TECHNISCHE
UNIVERSITÄT
DARMSTADT

ULB

Bidirectional Human-Robot Learning: Imitation and Skill Improvement

Sousa Ewerton, Marco Antonio

(2020)

DOI (TUprints): https://doi.org/10.25534/tuprints-00011875

Lizenz: 

CC-BY-SA 4.0 International - Creative Commons, Attribution Share-alike

Publikationstyp: Ph.D. Thesis

Fachbereich: 20 Department of Computer Science

Quelle des Originals: https://tuprints.ulb.tu-darmstadt.de/11875

Bidirectional Human-Robot Learning: Imitation and Skill Improvement

Bidirektionales Mensch-Roboter-Lernen: Nachahmung und Verbesserung der Fähigkeiten
Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von Marco Antônio Sousa Ewerton aus São Luís - MA, Brasilien
Tag der Einreichung: 28. Mai 2019, Tag der Prüfung: 15. Juli 2019
Darmstadt — D 17

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Masaki Takahashi

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Bidirectional Human-Robot Learning: Imitation and Skill Improvement
Bidirektionales Mensch-Roboter-Lernen: Nachahmung und Verbesserung der Fähigkeiten

Genehmigte Dissertation von Marco Antônio Sousa Ewerton aus São Luís - MA, Brasilien

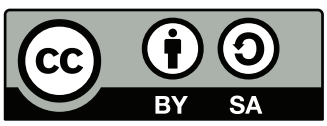
1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Masaki Takahashi

Tag der Einreichung: 28. Mai 2019
Tag der Prüfung: 15. Juli 2019

Darmstadt — D 17

Bitte zitieren Sie dieses Dokument als:
URN: urn:nbn:de:tuda-tuprints-118752
URL: http://tuprints.ulb.tu-darmstadt.de/id/eprint/11875

Dieses Dokument wird bereitgestellt von tuprints,
E-Publishing-Service der TU Darmstadt
http://tuprints.ulb.tu-darmstadt.de
tuprints@ulb.tu-darmstadt.de



Die Veröffentlichung steht unter folgender Creative Commons Lizenz:
Namensnennung – Weitergabe unter gleichen Bedingungen 4.0 International
http://creativecommons.org/licenses/by-sa/4.0/deed.de

Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 23. Juni 2020

(Marco Antônio Sousa Ewerton)

Abstract

A large body of research work has been done to enable robots to learn motor skills from human demonstrations. Likewise, much work has been done on investigating how humans can learn from robots or get physically assisted by them. However, these two bodies of work have been mostly detached from each other. A set of works in robotics has been enabling robots to learn motor skills through imitation and self-improvement while another set of works in robotics and sports science has studied how robots can help humans perform movements with less effort, assist humans in rehabilitation and in motor skill learning.

In most of the work on humans being assisted by robots, there is no attempt to improve upon predefined movements or demonstrations through machine learning techniques. In part, this lack of machine learning in the area of robot-assisted human motion is due to the high cost of collecting data of humans interacting with robots. Collecting this type of data might involve risks to humans, usually takes a large amount of time, usually requires expensive equipment and trained personnel. Collecting data of interactions between humans, from videos for example, does not present some of these difficulties. Nevertheless, data of interactions between humans may not be readily usable for machine learning techniques applied to robotics because of the differences between the morphology of humans and robots, lack of information about forces and torques, which may be critical to succeed in some tasks, endless variability of interaction scenarios, etc. Consequently, it is hard to collect enough data to infer the behavior of the robot given the behavior of the human and the environment in any possible situation. It is also hard to make the robot improve its interaction with humans through trial and error in a safe manner within a feasible amount of time. Therefore, even state-of-the-art work on robots assisting humans often avoids machine learning. Robot-assisted training and rehabilitation are based for example on spring-damper systems with fixed stiffness and damping coefficients. Assisted teleoperation relies for example on expert demonstrations or on rules designed to assist users in specific situations such as grasping multiple objects.

However, in order to increase the usability and range of applications of robot assistants, it is crucial that these robots autonomously adapt to different humans, tasks and environments. Moreover, it would be desirable that a robot could teach or assist a human just like a human can teach a new skill to a robot via kinesthetic teaching, i.e., moving the robot arm with gravity compensation.

Building on data-efficient machine learning techniques, this thesis presents a unifying perspective to the fields of robots learning from humans and humans from robots. Using stochastic movement representations and reinforcement learning techniques, we enable robots to learn motor skills from a few human demonstrations and improve these motor skills. Subsequently, the robot can help humans learn a new motor skill or perform a challenging task such as teleoperation requiring obstacle avoidance and accuracy.

We start by tackling one of the key problems in human-robot interaction: how can a robot learn to react to human movements which can vary both in shape and speed? We present a method to enable robots to learn interaction models from demonstrations with different speeds even if the demonstrations are corrupted by noise or occlusion, for example. These interaction models, which are based on stochastic movement representations, can then be used to predict the rest of a human action given observations of its beginning as well as to compute the most suitable robot reaction.

Often, human demonstrations are suboptimal because the human is a non-expert, the environment changes rendering the demonstrations ineffective, the robot cannot accurately reproduce the demonstration, etc. In such cases, it is necessary to improve upon the initial demonstrations. We present an approach to deal with this problem by enabling the human to teach new motor skills to robots through demonstrations and incremental feedback. Furthermore, by using probabilistic conditioning, the robot is able to generalize learned movements to different situations.

Human demonstrations can be suboptimal not only with respect to their shape in space but also with respect to their speed profile. The robot may have for example to adapt the speed of a movement to throw an object further away or closer, hit an object faster or slower, etc. These can be local adaptations of the speed profile instead of simply accelerating or decelerating the whole movement uniformly. Moreover, if a high degree of accuracy is required, it may be very difficult for a human to correct the initial demonstrations through incremental feedback. Besides that, it is desirable that the robot improve its movements without always requiring human input. We address this problem by optimizing through reinforcement learning movement parameters that determine the speed profile of movements.

Having addressed key problems towards making robots more capable of learning from human demonstrations and improve upon these demonstrations, we start to look into how robots can help humans learn new motor skills or perform a challenging teleoperation task. As before, using stochastic movement representations, we address the problem of giving visual feedback to a human trying to learn a new motor skill. Our proposed algorithm aligns expert demonstrations in space and time and builds a probability distribution of trajectories. When a user tries to perform that motor skill,

our algorithm uses the built probability distribution based on the expert demonstrations to evaluate if the user attempt matches or not the expert movements and gives visual feedback to the user.
A similar principle can be used to give haptic feedback to the user, helping him/her to succeed in challenging teleoperation tasks. As we demonstrate through user studies, teleoperation tasks may be difficult for humans when, for example, the human cannot accurately estimate the 3D position of objects or the human needs to control several degrees of freedom at the same time. In this case, the failed attempts of the human are used as initial demonstrations and are optimized through reinforcement learning to generate trajectory distributions that satisfy the requirements of the task.
We assist humans in teleoperation tasks initially by tackling motion planning problems in static environments. Subsequently, we build up a framework that enables robots to solve motion planning problems and interact with humans in dynamic environments. Finally, we tackle tasks with multiple possible solutions such as grasping multiple objects and infer the intention of the user to select the most appropriate guidance.
In summary, this thesis uses stochastic movement representations to enable robots to learn motor skills from human demonstrations and incremental feedback. These movement representations are used in conjunction with reinforcement learning algorithms to make the robot improve upon the demonstrations or upon arbitrary initial trajectory distributions. The robot can use trajectory distributions based on expert demonstrations to help non-experts acquire a new motor skill. In addition, if the human demonstrations are not suitable, the robot can use trajectory distributions optimized trough reinforcement learning to help users perform challenging tasks, e.g. in teleoperation. Further applications of the algorithms and frameworks proposed in this thesis may lie in fields such as rehabilitation, training in sports and human-robot collaboration.

Zusammenfassung

Zum einen wurde viel Forschung mit dem Ziel betrieben, Roboter dazu zu befähigen anhand menschlicher Demonstrationen motorische Fertigkeiten zu erlernen. Zum anderen wurde umfangreich erforscht, wie Menschen von Robotern lernen oder von ihnen physisch unterstützt werden können. Diese beiden Forschungsfelder sind jedoch weitestgehend voneinander getrennt. Eine Reihe von Arbeiten in der Robotik hat Roboter dazu in die Lage versetzt motorische Fertigkeiten anhand von Demonstrationen und durch Selbstverbesserung zu erlernen. Eine andere Reihe von Arbeiten in der Robotik und den Sportwissenschaften hat erforscht, wie Roboter Menschen dabei helfen können, Bewegungen mit geringerem Kraftaufwand auszuführen und wie Roboter Menschen bei der Rehabilitation sowie beim Erlernen motorischer Fertigkeiten unterstützen können.

In den meisten Arbeiten, die sich mit der Unterstützung des Menschen durch Roboter beschäftigen, wird nicht versucht vordefinierte Bewegungen oder Demonstrationen durch Techniken des maschinellen Lernens zu verbessern. Der mangelnde Einsatz von Techniken des maschinellen Lernens im Bereich der von Robotern unterstützten menschlichen Bewegung ist zum Teil auf die Tatsache zurückzuführen, dass die Datensammlung in Hinblick auf die Interaktion von Menschen mit Robotern sehr kostenintensiv ist. Eine Datensammlung dieser Art könnte mit Risiken für den Menschen verbunden sein. Darüber hinaus ist ein solches Vorhaben für gewöhnlich sehr zeitintensiv und setzt in der Regel eine kostspielige Ausstattung sowie geschultes Personal voraus. Die Erhebung zwischenmenschlicher Interaktionsdaten, beispielsweise mithilfe von Videos, ist mit einzelnen der zuvor genannten Schwierigkeiten nicht verbunden. Solche Interaktionsdaten sind jedoch nicht ohne weiteres verwendbar, wenn Techniken des maschinellen Lernens auf Roboter angewendet werden sollen. Schwierigkeiten bestehen zum einen aufgrund der unterschiedlichen Morphologie des Menschen und des Roboters. Zum anderen liegt ein Mangel an Informationen hinsichtlich der Kräfte und Drehmomente vor, was sich bei der Ausführung mancher Aufgaben als kritisch erweisen könnte, und es gibt eine unendliche Variabilität an Interaktionsszenarien. Folglich gestaltet sich die Erhebung einer ausreichenden Datenmenge, anhand derer in jeder möglichen Situation vom beobachteten Verhalten des Menschen und den Umweltbedingungen auf das Verhalten des Roboters geschlossen werden kann, als äußerst schwierig. Eine weitere Herausforderung besteht darin, auf sichere Art und Weise und innerhalb einer vertretbaren Zeitspanne, die Interaktion des Roboters mit dem Menschen mithilfe der Trial-and-Error-Methode zu verbessern. Aus diesem Grund werden in den meisten aktuellen Forschungsarbeiten, in denen es darum geht, wie Roboter Menschen unterstützen können, Techniken des maschinellen Lernens nicht in Erwägung gezogen. Arbeiten, die erforschen wie Roboter Menschen beim Training und bei der Rehabilitation unterstützen können, beruhen zum Beispiel auf Feder-Dämpfer-Systemen mit vorher festgelegten Steifigkeits- und Dämpfungs-Koeffizienten. Weiterhin basieren von Robotern unterstütze Teleoperationen zum Beispiel auf Experten-Demonstrationen oder auf Regeln, die dazu entwickelt wurden, Anwender in spezifischen Situationen, beispielsweise beim Greifen mehrerer Objekte, zu unterstützen. Um die Brauchbarkeit von Assistenz-Robotern zu verbessern und deren Anwendungsgebiet auszuweiten, ist es dennoch notwendig, dass sich diese selbstständig an verschiedene Menschen, Aufgaben und Umweltbedingungen anpassen. Darüber hinaus wäre es wünschenswert, wenn ein Roboter einem Menschen in gleichem Maße neue Dinge beibringen oder ihn unterstützen könnte, wie ein Mensch einem Roboter durch Kinesthetic Teaching neue Dinge beibringen oder ihn unterstützen kann. Mithilfe des Kinesthetic Teaching kann ein Roboterarm mit Ausgleich der Schwerkraft bewegt werden.

In der vorliegenden Thesis wird erforscht, wie dateneffiziente Techniken des maschinellen Lernens in wechselseitigen Lernprozessen zwischen Menschen und Robotern eingesetzt werden können. Durch die Anwendung von stochastischen Bewegungsrepräsentationen und Techniken des Reinforcement Learnings, befähigen wir Roboter dazu anhand einer kleinen Anzahl von menschlichen Demonstrationen motorische Fertigkeiten zu erlernen und diese Fertigkeiten zu verbessern. Danach kann der Roboter Menschen dabei helfen eine neue motorische Fertigkeit zu erlernen oder eine herausfordernde Aufgabe, wie eine Teleoperation, die die Umrundung von Hindernissen und Sorgfältigkeit erfordert, zu bewältigen.

Zunächst gehen wir eine der Hauptherausforderungen in der Mensch-Roboter-Interaktion an: Wie kann ein Roboter lernen auf menschliche Bewegungen zu reagieren, welche sowohl in ihrer Form wie auch in ihrer Geschwindigkeit variieren? Wir beschreiben eine Methode, die einen Roboter dazu befähigen kann, Interaktionsmodelle anhand von Demonstrationen unterschiedlicher Geschwindigkeit zu erlernen. Dabei ist es nicht von Bedeutung, ob die Demonstrationen beispielsweise durch Geräusche oder optischen Barrieren gestört werden. Diese Interaktionsmodelle, welche auf stochastischen Bewegungsrepräsentationen basieren, können anschließend dazu verwendet werden den Rest einer menschlichen Bewegung vorherzusagen, vorausgesetzt es liegen Beobachtungen zu deren Beginn vor, und die passendste Reaktion des Roboters zu berechnen. Häufig sind die Demonstrationen des Menschen suboptimal, da der Mensch kein Experte ist. Weitere Probleme entstehen dadurch, dass sich die Umweltbedingungen verändern und der Roboter nicht immer ganz akkurat die Demonstrationen reproduzieren kann. In einem solchen Fall ist es notwendig die Bewegungen der Anfangsdemonstration zu verbessern. Wir beschreiben einen Ansatz, mit dessen Hilfe diesem Problem begegnet werden kann. Mit diesem Ansatz befähigen wir den Menschen dazu, dem Roboter mithilfe von Demonstrationen und inkrementel-

lem Feedback neue motorische Fertigkeiten beizubringen. Darüber hinaus wird der Roboter mithilfe von probabilistischer Konditionierung dazu in die Lage versetzt gelernte Bewegungen auf verschiedene Situationen zu generalisieren.
Menschliche Demonstrationen können sich nicht nur in Hinblick auf die Form, sondern auch in Hinblick auf das Geschwindigkeitsprofil als suboptimal erweisen. Zum Beispiel könnte der Fall eintreten, dass der Roboter die Geschwindigkeit einer Bewegung anpassen muss, um ein Objekt weiter oder weniger weit zu werfen, ein Objekt schneller oder langsamer zu treffen, etc. Im Zuge dessen könnte eine einheitliche Beschleunigung oder Verlangsamung der ganzen Bewegung nicht ausreichend sein und eine lokale Anpassung des Geschwindigkeitsprofils notwendig werden. Wenn ein hoher Grad an Exaktheit notwendig ist, könnte es darüber hinaus für einen Menschen sehr schwierig sein, die ursprüngliche Demonstration mithilfe von inkrementellem Feedback zu verbessern. Zusätzlich wäre es wünschenswert, dass der Roboter seine Bewegung selbstständig verbessert, ohne fortwährend ein Input des Menschen zu benötigen. Wir gehen dieses Problem an, indem wir die Bewegungsparameter, die das Geschwindigkeitsprofil der Bewegungen bestimmen, mithilfe von Reinforcement Learning optimieren.
Nachdem wir einige Hauptprobleme adressiert haben, die dabei entstehen, wenn es darum geht Roboter dazu zu befähigen Bewegungen anhand menschlicher Demonstrationen zu lernen und diese Bewegungen zu verbessern, betrachten wir wie Roboter Menschen dabei helfen können neue motorische Fertigkeiten zu erlernen und herausfordernde Aufgaben der Teleoperation auszuführen. Auch an dieser Stelle verwenden wir stochastische Bewegungsrepräsentationen. Zunächst versuchen wir Menschen durch visuelles Feedback beim Erlernen neuer motorischer Fertigkeiten zu unterstützen. Unser vorgeschlagener Algorithmus justiert Experten-Demonstrationen in Hinblick auf Raum und Zeit und erstellt eine Wahrscheinlichkeitsverteilung der Trajektorien. Wenn ein Anwender versucht eine bestimmte Bewegung auszuführen, verwendet unser Algorithmus die erstellte Wahrscheinlichkeitsverteilung, welche auf den Experten-Demonstrationen basiert, um zu evaluieren ob der Versuch des Anwenders mit der Bewegung des Experten übereinstimmt oder nicht. Außerdem gibt unser Algorithmus dem Anwender visuelles Feedback.
Ein ähnliches Prinzip kann verwendet werden, um dem Anwender haptisches Feedback zu geben, das ihm oder ihr dabei helfen soll, herausfordernde Aufgaben der Teleoperation auszuführen. Wie wir in unseren Studien zeigen, können sich Aufgaben der Teleoperation für den Menschen als schwierig erweisen. Zum Beispiel kann der Mensch nicht exakt die 3D-Position von Objekten schätzen oder er hat Schwierigkeiten bei der gleichzeitigen Kontrolle von verschiedenen Freiheitsgraden. In diesem Fall werden die Fehlversuche des Menschen als Anfangsdemonstrationen verwendet und anhand von Reinforcement Learning verbessert. Mithilfe von Reinforcement Learning werden Trajektorien-Verteilungen, die den Anforderungen der Aufgabe gerecht werden, generiert.
Zunächst unterstützen wir Menschen bei der Durchführung von Aufgaben der Teleoperation indem wir Probleme des Motion Plannings in statischen Umgebungen angehen. Danach erarbeiten wir die Rahmenbedingungen, welche Roboter dazu befähigen Probleme des Motion Plannings und der Interaktion mit Menschen unter dynamischen Umweltbedingungen zu bewältigen. Schließlich behandeln wir Aufgaben mit verschiedenen möglichen Lösungen, wie zum Beispiel das Greifen unterschiedlicher Objekte, und leiten die Intention des Anwenders ab, um die passendste Anleitung auszuwählen.
Zusammenfassend lässt sich festhalten: Die vorliegende Thesis verwendet stochastische Bewegungsrepräsentationen, um Roboter dazu zu befähigen anhand menschlicher Demonstrationen und inkrementellem Feedback motorische Fertigkeiten zu erlernen. Diese Bewegungsrepräsentationen werden in Verbindung mit Reinforcement Learning-Algorithmen verwendet, um den Roboter dazu in die Lage zu versetzen die Anfangsdemonstrationen oder eine beliebige Trajektorien-Verteilung zu verbessern. Der Roboter kann Trajektorien-Verteilungen, die auf Experten-Demonstrationen basieren, verwenden, um Nicht-Experten dabei zu helfen neue motorische Fertigkeiten zu erlernen. Falls die menschlichen Demonstrationen nicht geeignet sind, kann der Roboter außerdem die durch Reinforcement Learning optimierte Trajektorien-Verteilung benutzen, um dem Anwender dabei zu helfen herausfordernde Aufgaben, zum Beispiel bei der Teleoperation, auszuführen. Weitere Anwendungen der Algorithmen und Frameworks, die in dieser Thesis vorgestellt werden, könnten in Feldern wie der Rehabilitation, des Sporttrainings oder der Zusammenarbeit von Mensch und Roboter liegen.

Acknowledgments

This thesis has only been possible due to the support of several people. Prof. Jan Peters has given me valuable research advice, many tips on time management and writing. Likewise, I have learned a lot from Guilherme Maeda. It has been a pleasure to work on several papers with Guilherme and exchange research ideas with him.
Furthermore, it has been a pleasure to work in an interdisciplinary project with Gerrit Kollegger and Prof. Josef Wiemeyer. The knowledge and experience of Gerrit and Josef have complemented mine. We have had a fruitful collaboration in the intersection between Sport Science and Robotics.
I want to express a big thanks to all the colleagues and to the staff of the Intelligent Autonomous Systems group of the TU Darmstadt with whom I have worked over the years. You all have contributed directly or indirectly to the results of this thesis.
I am thankful for having had excellent students who have taught me at least as much as I have been able to teach them.
I am grateful for having met Prof. Masaki Takahashi and his students. I will never forget the kindness of you all and look forward to keep our collaboration.
I would like to thank the members of my thesis committee, Professors Faust, Hollick, Kersting, Peters, Takahashi and Wiemeyer for investing the time to analyze my work. Your feedback is very welcome!
I am lucky for having a family that has always supported me and motivated me to improve myself and to be helpful to others.
Through my fiancée, Marie, I have gained a second family in Germany that is also very supportive and helps to make my life more enjoyable.
In special, I thank Marie, who has been the person who has accompanied me the closest, who has helped me and kept me motivated through the good and tough times. Marie, du bist die Beste!

Contents

Notation and Acronyms

Notation	Description
x	scalar
$\mathbf{x}$	vector
$\mathbf{X}$	matrix
$\mathbf{X}^\top$	transpose of a matrix
$\mathbf{X}^{-1}$	inverse of a matrix
$\{x_1, x_2, \cdots, x_n\}$	set of elements $x_1, x_2, \cdots, x_n$
$[x_1, x_2, \cdots, x_n]^\top$	column vector of elements $x_1, x_2, \cdots, x_n$
$\dot{x}$	time derivative
$\ddot{x}$	second derivative with respect to time
$p(x)$	probability density function
$p(x\|y)$	conditional probability density function of x given y
$\log(x)$	natural logarithm
$\mathbf{I}_{N\times N}$	identity matrix with N rows and N columns
$\|\|x\|\|$	L^2 norm
$\mathbb{E}_{\boldsymbol{\theta}}(x) = \int x p_{\boldsymbol{\theta}}(x) dx$	expected value according to probability density function with parameters $\boldsymbol{\theta}$

Acronym	Description
DTW	Dynamic Time Warping
EM	Expectation Maximization
GMM	Gaussian Mixture Model
GP	Gaussian Process
HMM	Hidden Markov Model
PRO	Pearson-Correlation-Based Relevance Weighted Policy Optimization
ProMP	Probabilistic Movement Primitive
RWPO	Reward Weighted Policy Optimization
RWR	Reward-weighted Regression
VIPS	Variational Inference by Policy Search

1 Introduction

There is a large body of work in enabling robots to learn from human demonstrations, a research field known as imitation learning [1, 2]. Whether and what humans can learn from robots have also been intensively researched subjects [3–6]. However, in most approaches for humans learning or getting assisted by robots, the robot movements are predefined or based on expert demonstrations, i.e., the robot does not improve upon the demonstrations through machine learning.
Rauter et al. [6] point “the missing big data” as one of the main issues hindering the application of machine learning in the field of human-robot interaction, especially in applications such as robot-assisted rehabilitation, surgery and human motor learning. The lack of big data in these applications is due to the fact that collecting human-robot interaction data may involve risks to the human, takes a substantial amount of time, requires costly equipment, etc.
Nevertheless, there exist machine learning techniques to extract patterns from relatively little data. Machine learning can be used for example to generate stochastic models of movements based on a few human expert demonstrations. Moreover, as long as it is possible to simulate the aspects of the environment that are important to succeed in a certain task, reinforcement learning can be used to optimize movements in that environment even if this optimization takes much more iterations than it would be feasible with real robots and real humans. In this thesis, we demonstrate how stochastic movement models and reinforcement learning techniques can be useful to address both robots learning from humans (imitation learning) and humans learning from or getting assisted by robots.
One of the possible applications of our work is assisted teleoperation. A robot may be trained through demonstrations to assist a human in teleoperation. Training through demonstrations is appealing because it is less time intensive than manually defining virtual fixtures [7]. However, there are many situations where the human demonstrations that are available are suboptimal or where the motor skills need to be adapted to environments very different from the ones in which demonstrations were given. Therefore, it is desirable that the robot improve upon suboptimal demonstrations or unsuccessful human attempts to solve a task and adapt to help the human by using haptic feedback.
Previous research provides evidence of the importance of haptics in learning motor skills [8–10]. This evidence is an indication that the approach presented in this thesis may also have potential for motor skill learning through haptic feedback.
In this thesis, demonstrations may be considered suboptimal due to three reasons. One reason is that the demonstrator is not an expert, and therefore his/her demonstrations do not satisfy a certain criterion of optimality. Another reason is that, although a certain demonstration may satisfy the optimality criteria, it is possible that the robot cannot generate the forces and torques to reproduce that demonstration, rendering it effectively suboptimal. Finally, changes in the environment may turn a demonstration "obsolete". For example, if demonstrations were carefully provided on a cluttered environment to avoid collisions, they are optimal with respect to a collision cost. However, if objects change position such that the path described by the demonstration is now in collision with some objects, the demonstration is not optimal for the new scene. This thesis proposes optimizing such suboptimal demonstrations by using a reinforcement learning approach. That means an explicit formulation of a cost/reward is provided, and the robot attempts to optimize the suboptimal demonstrations by trial and error. This optimization also opens the opportunity for the robot to transmit this information back to the demonstrator, in which haptic feedback is one of the most natural and practical ways.
We demonstrate applications of our framework to human-robot collaboration, motor-skill learning, teleoperation and planning in dynamic environments. The approach here presented has also potential in rehabilitation and human learning from haptic feedback.
In summary, this thesis presents a new and unifying outlook on how robots can learn from humans and how humans can learn or get assisted by robots. The same framework that is used to let robots learn motor skills from human demonstrations and to adapt these demonstrations to specific requirements is also used to help the human learn motor skills or to assist the human in shared control tasks. Fig. 1.1 illustrates the problem addressed in this thesis, our approach and some of our experiments.

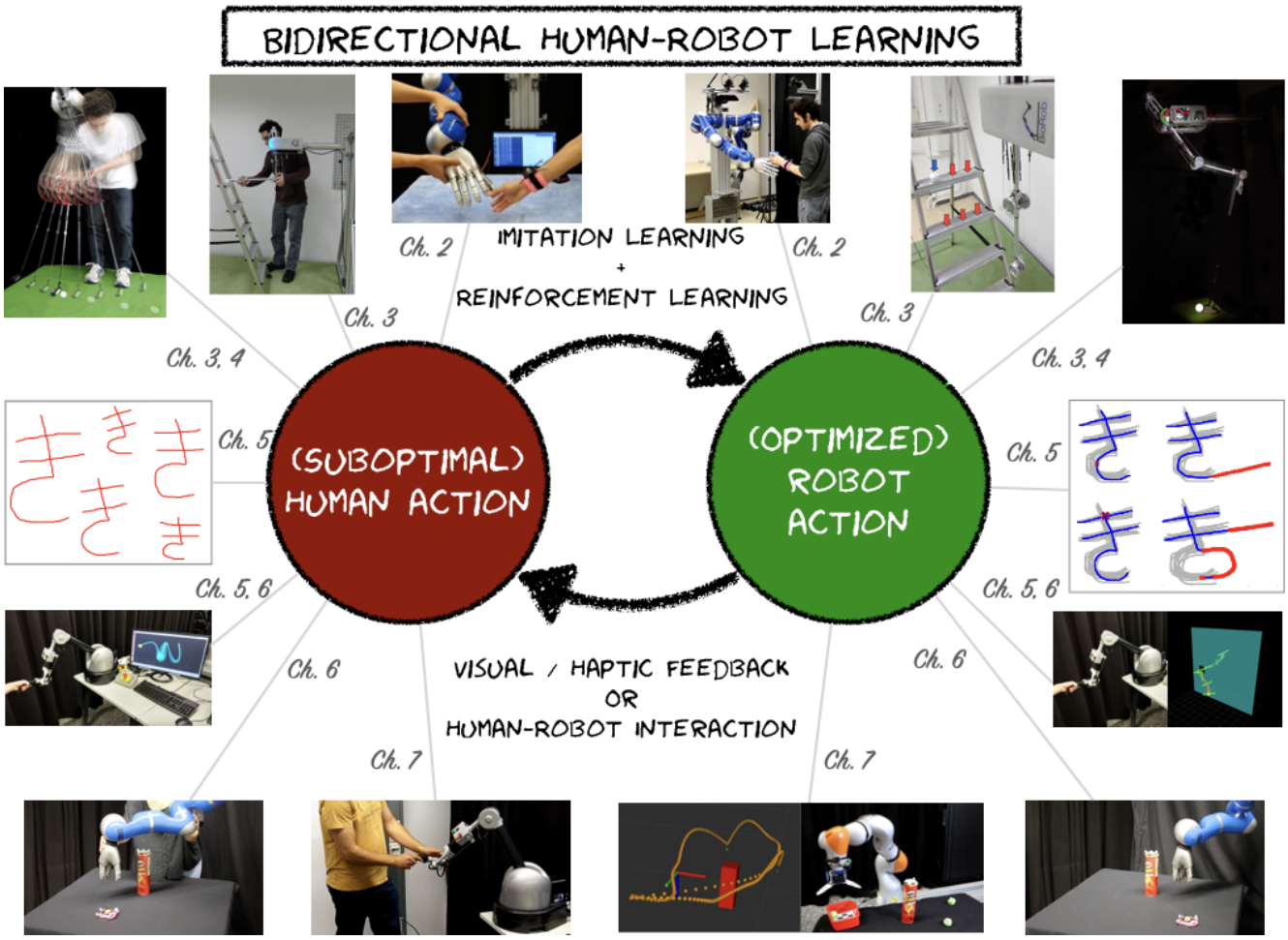


Figure 1.1.: This thesis addresses the "Bidirectional Human-Robot Learning" problem: How can humans and robots learn from each other? Humans can teach new motor skills to robots through demonstrations (imitation learning). These demonstrations may be suboptimal for several reasons: the human may be a non-expert or incapable of providing optimal demonstrations; the environment may change, rendering the demonstrations ineffective; the robot may be incapable of reproducing the provided demonstrations. In these cases, reinforcement learning can be used to optimize the robot movements. The optimized robot movements can in turn be used to teach or assist a human through visual or haptic feedback and to enable other forms of human-robot interaction as well. This figure presents several images that are representative of our work along with the corresponding thesis chapters. In Chapter 2, a robot learns from demonstrations how to react to human actions executed at different speeds. In Chapter 3, a robot learns a reaching task and a golf putt from incremental human demonstrations. In Chapter 4, the robot uses reinforcement learning to adapt the speed profile of the movement originally learned from human demonstrations. In Chapter 5, visual feedback can be provided to a user learning a motor skill (in this case, Japanese characters). The visual feedback is based on a probability distribution of trajectories that are aligned by our algorithm in space and time. Also in Chapter 5, haptic feedback based on optimized movements helps a user in an assisted teleoperation task. In Chapter 6, the robot uses reinforcement learning to generalize movements learned from human demonstrations to different situations and to adapt to dynamic environments. Finally, in Chapter 7, reinforcement learning finds multimodal solutions to assist users in teleoperation tasks. The assistive framework presented in that chapter adapts to the actions of the human operator and to changes in the environment.

1.1 Motivation

This thesis is motivated by the "Bidirectional Human-Robot Learning" problem. It means that both agents, human and robot, should learn from each other and profit from each other's capabilities.
More precisely, we address the following questions: How can robots learn from humans? How can humans learn or get assisted by robots? How can a solution to a human-robot interaction task, e.g. robot-assisted teleoperation or human-robot shared control, be found even if both human and robot are unable in the beginning to provide acceptable solutions? How can a robot optimize and generalize human demonstrations as well as trajectories based on prior human knowledge to successfully solve tasks and collaborate with the human in dynamic environments?
The answers to these questions may have implications for several possible uses of robots, such as helping to lift and to move patients in hospitals, assisting people in the household, guiding users in public places, etc. In the following, several applications of robots related to "Bidirectional Human-Robot Learning" are discussed.

Human-Robot Collaboration in the Workplace

Previous studies have established links between overhead work, where workers need to raise one or both arms above their shoulders, and upper extremity discomfort and disorders [11]. There is also strong evidence for a causal relationship between forceful or highly repetitive work and musculoskeletal disorders (MSDs) involving the neck or the neck and the shoulders [12]. Moreover, it has been demonstrated that work-related lifting and forceful movements contribute to low-back disorders [12]. A recent meta-review confirms that finding: "Moderate to high-quality evidence is available that LRS (lumbosacral radiculopathy syndrome) can be classified as a work-related disease depending on the level of exposure to bending of the trunk or lifting and carrying. Professional driving and sitting were not significantly associated with LRS" [13]. Knee diseases may as well be caused by work-related movements: "There is moderate quality evidence that longer cumulative exposure to kneeling or squatting at work leads to a higher risk of osteoarthritis of the knee" [14].
Robots can relieve humans of unergonomic tasks in the workplace such as overhead work. Robots can also lift heavy loads and perform repetitive tasks, reducing the risk of injuries. Therefore, human-robot collaboration has great potential to improve the quality of life of human workers.
Human-robot collaboration has been finding more and more applications in the industry [15]. The company KUKA, for example, lists on its website several applications of their robots to human-robot collaboration in the industry [16].
The use of human-robot collaboration in the industry is increasing due to "advances in sensors, hydraulics, mobility, machine vision and big data" [17]. Despite these technological advances, human-robot interaction is still far from being as seamless as human-human interaction. This thesis contributes towards that goal by presenting algorithms that endow robots with (1) the ability of learning new tasks, including tasks which involve human-robot interaction, from human demonstrations, (2) flexibility to adapt to human movements with different shapes and speed profiles during human-robot interaction, (3) self-improvement capabilities to optimize upon suboptimal demonstrations and to adapt to new situations.

Robot-Assisted Surgery

Robots have been used to enhance the precision of surgeons by presenting them with a "magnified 3D high-definition vision" and by scaling and filtering their hand movements [18, 19]. Virtual fixtures [7] and human-machine cooperative systems have also been applied to robot-assisted surgery [20]. For example, a "safety barrier or *no-fly zone*" is determined given registered anatomical models. A safety barrier prevents the robot's tools from entering certain regions of the robot's workspace. The robot can also help the surgeon maintain a desired force or adjust the position and the orientation of a tool [20].

Robot-Assisted Neurorehabilitation

Studies on the effectiveness of robot-assisted neurorehabilitation often report inconsistent results in part due to the difficulty in performing statistically significant studies in this field, since treatment must be suited to the necessities of each patient. In fact, inconsistencies are also present in conventional neurorehabilitation training. However, according to many studies, robots can improve rehabilitation outcome [21]. Adaptive assistance properties have been identified as important features for robots in neurorehabilitation [22]. Iosa et al. [21] highlight the potential of artificial intelligence to automatically adjust the parameters of the assistive robot given the requirements of the human therapist. Nowadays, the human therapist may need to tune several robot parameters to change for example the walking speed of the patient.

Robots Assisting People with Disabilities

Muelling et al. [23] propose a framework to help users teleoperate a robot through Brain-Computer Interfaces (BCI). Their framework includes computer vision techniques to recognize and localize objects, user intention inference and arbitration between control by the user and control by the robot depending on the certainty about the user intention. Our work could potentially add up to [23] in different ways. The framework proposed in [23] relies on a library of preprogrammed grasp poses and movements for tasks such as door opening and pouring soda. An approach based on learning from demonstrations and reinforcement learning such as the one proposed in this thesis could potentially be able to learn these poses and movements. Our work also provides a way to quickly compute distributions of trajectories that avoid obstacles and reach targets in dynamic environments. The computed distributions of trajectories could then be used to assist the user. The input of the user would determine which objects should be grasped and which objects should be treated as obstacles, for example, using the intention inference algorithm proposed in [23]. The inferred intention could in turn be used to compute a probability distribution of trajectories to solve the task.
Several other examples of robots to support people with disabilities in daily tasks as well as rehabilitation robots are provided in [24].

Teleoperation

The first master-slave teleoperator was built in the mid-1940s [25]. Today teleoperation is used by the military to investigate potentially explosive devices, in radioactive environments, underwater, for space exploration, etc. and it continues to be an active topic of research [26–28].
Training robots to assist humans in teleoperation tasks through demonstrations is a promising alternative to manually defining guiding virtual fixtures [7]. Training through demonstrations is intuitive for the human user and not very time intensive. In [26], a robot learns from demonstrations to assist the user with the teleoperation of a device to scan a surface, such as in a maintenance task performed by remotely operated underwater vehicles (ROVs). In [27], the robot learns assistive behavior for teleoperation tasks from iterative refinement by the human.
Learning from demonstrations has also been applied in supervisory control [28]. In this paradigm, the remote system executes a task autonomously, while the user needs only to specify high-level task goals.
In contrast to these methods, in our work, the robot optimizes, through trial and error, upon initial demonstrations or human attempts to solve a task. The optimized robot movement can in turn be used to assist the human. In addition, by carefully designing a reward function that attributes higher values to desirable movements and lower values to undesirable ones, it may be possible to find a suitable assistive behavior in teleoperation tasks without the need of human demonstrations. Learning assistive teleoperation without demonstrations is discussed in Chapter 7.

Robot-Assisted Training in Sports

Kümmel et al. [10] show evidence that humans can learn sport-specific motor skills through robot guidance. In the user studies conducted in that work, the robot guides the human several times along the same predefined trajectory. The motor skill examined in that work was a golf swing.
Robots that help users learn golf swings have been employed by golf schools since 2014 [29,30]. A report extracted from "forbes.com" [30] briefly explains the functionality of such a robot:
"(RoboGolfPro) It's a robotic golf-training system that stores pre-recorded PGA Tour pro swings in it. Any amateur can grab onto the grip of a club that's connected at the other end to a robot, and then be guided through a pro's swing – right down to every detail and at least at first in slow motion. "
"A capable golf instructor trained on the software can build any golfer an ideal swing based off his or her physical dimensions, range of motion, current swing and swing orientation."
Still in this same report, a golf instructor points out an improvement in the process of teaching students who don't speak English because the instructor is able to teach through the robot certain aspects of the movement that he could not communicate through words to the student.
There is thus great potential for the application of robots in sports training. It may be possible that machine-learning-based approaches such as the one adopted in this thesis contribute to this kind of robot application by automatizing the search for optimal movements given certain requirements and using the solution to this optimization problem to give the human visual and haptic feedback.
Rauter et al. [6] present an approach for robot-assisted motor skill learning with automated feedback selection. The motor skill studied in [6] is a rowing movement. The proposed framework can automatically select a feedback type for a user depending on the dominant error made by that user. This personalized feedback has a significant effect on reducing the velocity error. Differently from our work, that work does not address the learning of trajectories and velocity profiles.

Numerous other works discuss possible applications of robots or autonomous systems to training in sports: automated anomaly detection based on correct demonstrations [31]; visual feedback based on correct demonstrations [32]; visual, auditory and haptic feedback modalities [33].

Exoskeleton Robots

In [34], reinforcement learning is used to make a 1-DoF exoskeleton robot learn assistive strategies from user-robot physical interaction. That work is very much in line with this thesis since it aims at achieving an optimal movement for the human in interaction with the robot starting from suboptimal movements. Due to the necessity of optimizing with the human in the loop, the robot needs to have few degrees of freedom or synergies between its degrees of freedom must be explored, allowing for control in a lower dimensional space. Otherwise, the number of iterations to achieve an optimal movement becomes very high, rendering the optimization with the human in the loop infeasible.
In Chapter 6, the robot optimizes its movements offline, using the information about the environment that is necessary to solve the task at hand. Subsequently, this optimized movement is used to help the human perform a certain movement. Perhaps an interesting line of research would be to combine both approaches, using the optimization iterations with the human in the loop to update a model of how the human reacts to the assistance by the robot. This model could in turn be used to perform several optimization iterations without requiring the interaction with the human. This approach could potentially make feasible optimization of human movements assisted by robot exoskeletons with a fair amount of degrees of freedom.

1.2 Contributions

Humans are still much more versatile than robots when it comes to learning and applying motor skills in unstructured environments. On the other hand, robots can perform very accurate measurements of positions, speeds, accelerations, forces and torques, which might be useful for humans trying to learn or optimize movements. Based on this perspective, the work presented in this thesis has studied how humans and robots can better interact with each other and learn motor skills from each other.

We have developed algorithms that enable people to teach robots new motor skills through demonstrations instead of writing several lines of code. These algorithms allow the robot to learn activities such as reaching objects in its workspace, but also how to interact with humans moving at different speeds (Chapter 2). Furthermore, the algorithms presented in that chapter enable the robot to learn motor skills from demonstrations with missing data, which can result from occlusions or lack of sensor coverage when recording demonstrations.

When learning from demonstrations, the robot must be able to not only imitate the human but also to generalize his/her movements to new situations. Besides, it might be useful to allow the human to give incremental feedback to the robot. For this reason, we have developed an algorithm for incremental imitation learning of context-dependent motor skills (Chapter 3).

Incremental imitation learning can be a fast way of teaching new motor skills to robots when the human is able to incrementally correct the movements of the robot. However, in several cases, the human lacks the accuracy to perform fine adjustments to a given movement. Making use of the ability of the robot of performing precise measurements, we have developed algorithms to enable the robot to optimize a movement learned from human demonstration in order to achieve precise speeds that are different from the ones of the original demonstrated movement (Chapter 4).

Building upon stochastic movement representations that play a key role in our work on imitation and reinforcement learning, we demonstrate how a machine can create a probabilistic model of a certain motor task given expert demonstrations, which allows it to give feedback to a human trying to learn how to solve that task. Moreover, through reinforcement learning, the initial probabilistic model can be optimized to help humans succeed in a shared control task such as robot-assisted teleoperation even when the available demonstrations are insufficient (Chapter 5).

The framework proposed in Chapter 5 can assist the human in static environments. Chapter 6 improves and extends this framework to solve tasks in dynamic environments. The results of this chapter have potential applications in robot-assisted teleoperation and other forms of human-robot interaction where it is important that the robot constantly adapt to the movements of the human partner and to other changes in the environment.

In order to effectively assist humans in teleoperation tasks involving several obstacles and several objects of interest, it is important to identify multiple possible solutions. If none of the proposed solutions correspond to the real intention of the human, it is important to enable the human to escape the guidance of the autonomous system, in which case the autonomous system computes a new set of solutions to correspond to the intention of the human. A framework to address these challenges is proposed in Chapter 7.

In summary, this thesis gradually builds up a framework to teach robots new motor skills from human demonstrations, optimize, adapt these motor skills and use the optimized motor skills to help humans. In this framework, expert demonstrations or optimized movements can be used to help humans learn motor skills or to help humans through shared autonomy in tasks such as robot-assisted teleoperation. Moreover, the proposed framework can be used to address human-robot collaboration. In contrast to most previous works addressing the learning of shared autonomy tasks from demonstrations or addressing human-robot interaction, our work addresses the problem of learning motor skills when both human and robot are, at first, unable to perform successful movements or when it is necessary to deal with situations very different from situations for which there were demonstrations.

Addressing both robots learning from humans and humans learning from or getting assisted by robots when both agents are potentially unable to solve the task at hand by themselves is what we call the “Bidirectional Human-Robot Learning” problem. Probability distributions of trajectories and reinforcement learning play a key role in our approach to solve this problem. Fig. 1.2 shows a schematic representation of this problem and of our proposed approach.

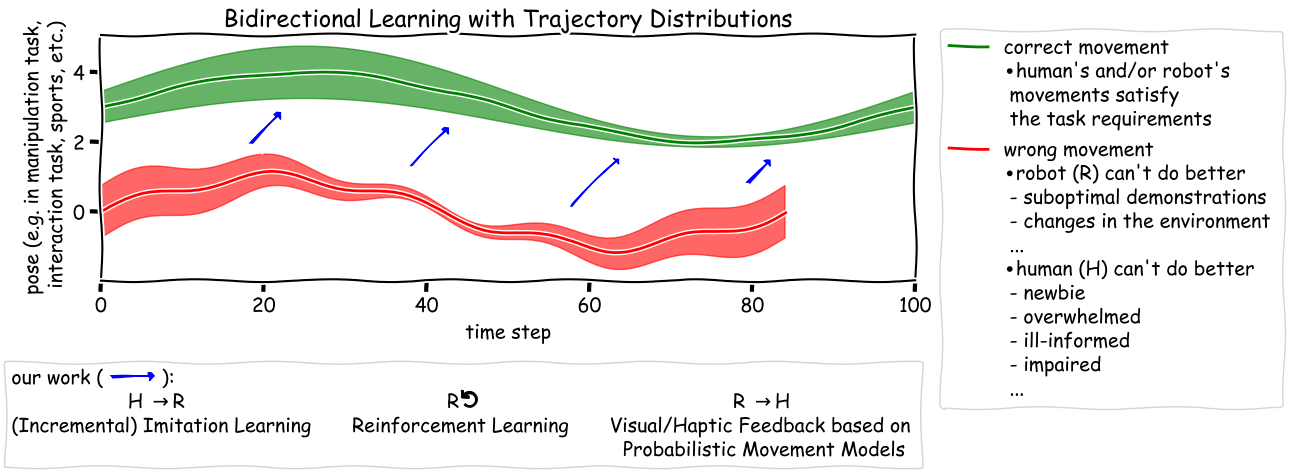


Figure 1.2.: Addressing the "Bidirectional Human-Robot Learning" problem with trajectory distributions. Trajectory distributions play a key role in this thesis. Trajectories may be wrong with respect to the requirements of the task at hand because they do not have an adequate shape in space or do not have an adequate speed profile, e.g. the wrong trajectories are too fast such as in this figure. Our work models movements as trajectory distributions and presents algorithms to optimize these distributions. Trajectories provided by the human may be wrong for several reasons: the human may be a non-expert, he/she may be exposed to an overwhelming amount of sensor data related to the task at hand or the controls can be too complex, he/she may lack some important knowledge about the task at hand, he/she may be impaired in such a way that hinders the accomplishment of the task at hand, etc. The movements executed by the robot may be wrong because the robot may rely on suboptimal human demonstrations, the robot may be unable to produce the necessary forces and torques to reproduce human expert demonstrations or the environment can change, rendering the movements originally learned from human demonstrations ineffective. Our work addresses the improvement of motor skills in three ways: the human teaches motor skills to the robot, potentially in an incremental manner that enables the human to gradually improve the movements of the robot; the robot improves its own motor skills by trial and error; the robot uses movements learned from expert human demonstrations or optimized by the robot itself to teach a motor skill to a human or to assist the human in a shared-control task, e.g. teleoperation.

1.3 Outline of the Thesis

The order of the chapters in this thesis reflects the gradual development of our framework from addressing only imitation learning ($H \rightarrow R$) to imitation and reinforcement learning ($H \rightarrow R$ and $R \circlearrowleft$) up to the full bidirectional human-robot learning problem ($H \rightarrow R$, $R \circlearrowleft$ and $R \rightarrow H$), where both human and robot start without being able to solve the task at hand but learn from and interact with each other to arrive at a successful solution. The chapters can nevertheless be read independently as each chapter has been written in a self-contained manner. Table 1.1 provides an overview of the content of each chapter, the learning directions ($H \rightarrow R$, $R \circlearrowleft$ or $R \rightarrow H$) covered in each chapter and the corresponding research papers.

Chapter	H → R	$R \circlearrowleft$	$R \rightarrow H$
Ch. 2) Learning Motor Skills from Partially Observed Movements Executed at Different Speeds [IROS 2015]	✓		
Ch. 3) Incremental Imitation Learning of Context-Dependent Motor Skills [HUMANOIDS 2016]	✓		
Ch. 4) Movement Primitives with Multiple Phase Parameters [ICRA 2016]	✓	✓	
Ch. 5) Assisting Movement Training and Execution with Visual and Haptic Feedback [Frontiers 2018]	✓	✓	✓
Ch. 6) Learning Trajectory Distributions for Assisted Teleoperation and Path Planning [submitted to Frontiers 2019]	✓	✓	✓
Ch. 7) Assisted Teleoperation in Changing Environments with a Mixture of Virtual Guides [submitted to Advanced Robotics 2019]	✓	✓	✓

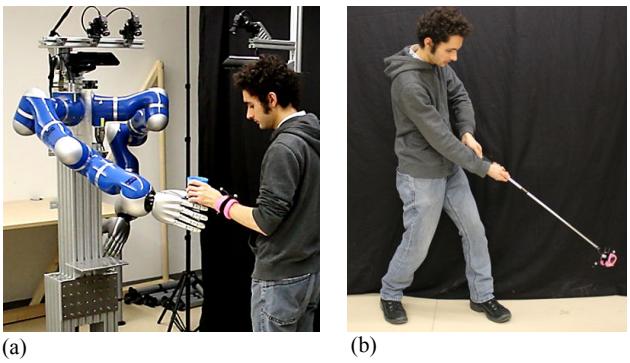
Table 1.1.: Overview of this thesis and how each chapter relates to each of the three learning directions investigated in this thesis: robots learning from humans ($H \rightarrow R$), robot self-improvement by trial and error ($R \circlearrowleft$) and humans learning or getting assisted by robots ($R \rightarrow H$). Chapter 2 addresses imitation learning when there is missing data in the demonstrations due to occlusions or lack of sensor coverage, for example. This chapter also addresses imitation learning of human-robot cooperation tasks when the robot needs to react to human movements executed at different speeds. Chapter 3 addresses a way to improve the robot movements: incremental human feedback. In Chapter 4, reinforcement learning is used to enable the robot to optimize by trial and error the speed profile of a movement. In Chapter 5, we show how stochastic movement representations and reinforcement learning can be used to teach and/or assist humans. Chapter 6 improves upon Chapter 5 by enabling adaptation of movements to dynamic environments. This adaptation is crucial in physical human-robot interaction and is promising for assisted teleoperation or shared control in dynamic environments. Finally, Chapter 7 presents a framework for assisted teleoperation in changing environments with multimodal solutions.

2 Learning Motor Skills from Partially Observed Movements Executed at Different Speeds

Learning motor skills from multiple demonstrations presents a number of challenges. One of those challenges is the occurrence of occlusions and lack of sensor coverage, which may corrupt part of the recorded data. Another issue is the variability in the speed of execution of the demonstrations, which may require a way of finding the correspondence between the time steps of the different demonstrations. In this chapter, an approach to learn motor skills is proposed that accounts both for spatial and temporal variability of movements. This approach, based on an *Expectation-Maximization* algorithm to learn *Probabilistic Movement Primitives*, also allows for learning motor skills from partially observed demonstrations, which may result from occlusion or lack of sensor coverage. An application of the algorithm proposed in this work lies in the field of Human-Robot Interaction when the robot has to react to human movements executed at different speeds. Experiments in which a robotic arm receives a cup handed over by a human illustrate this application. The capabilities of the algorithm in learning and predicting movements are also evaluated in experiments using a data set of letters and a data set of golf putting movements.

2.1 Prologue

Researchers investigating how a robot can learn motor skills from multiple human demonstrations face numerous challenges. The recorded data may, for example, be corrupted by occlusions or lack of sensor coverage and the demonstrations may vary considerably in the speed of execution. It is necessary to identify the corrupted data or to be able to extract useful information from the corrupted recordings. A common approach to represent the learned movement is to use time-dependent models. In this case, the variability in the speed of execution requires a way of finding the correspondence between the time steps of the different demonstrations. This chapter presents an approach to deal with both challenges. This approach is based on an Expectation-Maximization (EM) algorithm to learn Probabilistic Movement Primitives (ProMPs).



(a) (b)

Figure 2.1.: Experimental setups to evaluate our EM algorithm to learn ProMPs. (a) The robot receives a cup from a human who executes the handover at different speeds. (b) A golf putting data set is used to evaluate different formulations of our phase function.

ProMPs [35] are movement representations based on a probability distribution over trajectories. By representing movements with ProMPs, it is possible to condition the distributions over trajectories on observed positions. One of the applications of this framework lies in the field of Human-Robot Interaction, where it is possible to use probabilistic operations to condition the reactions of the robot on the actions of the human [36, 37].

This work builds upon the concept of ProMPs and proposes learning from multiple demonstrations probability distributions not only over trajectories but also over speed profiles or the phase of the movement. The phase of the movement is a function of time that can be associated with a movement. By changing the phase of a movement, it is possible to change its speed of execution. Learning a distribution over phase profiles is important to learn motor skills from demonstrations executed at different speeds and to allow a robot to react to human actions executed at different speeds.

The remainder of this chapter is organized as follows: Section 2.2 presents related work. Section 2.3 presents the main contribution of this work, an EM algorithm to learn ProMPs, capturing variability in the trajectories and in the phase profiles. Section 2.4 describes a number of experiments that evaluate the proposed algorithm. Our experiments evaluate the capabilities of the presented algorithm in learning movements from demonstrations with missing data. Furthermore, we evaluate two different formulations of the phase function: a simple formulation with one single phase parameter and a more general formulation with multiple phase parameters. Section 2.5 summarizes the chapter and presents ideas for future work.

2.2 Related Work

Time-dependent movement representations are heavily used due to their small number of parameters. The most prominent time-dependent movement representation is the Dynamical Movement Primitive (DMP) [38], which comprises a proportional-derivative (PD) controller and a nonlinear forcing function. This forcing function corresponds to a weighted sum of time-dependent basis functions. The weights for the basis functions are learned for example through linear regression.

In the Probabilistic Movement Primitive (ProMP), which is also a time-dependent movement representation, not a forcing function, but the trajectories themselves are approximated by a weighted sum of time-dependent basis functions.

In both approaches, the time-dependency of the basis functions can be given through a phase variable, which is in turn a function of time. Defining the basis functions through a phase variable allows for time-rescaling trajectories and synchronizing the movements of different limbs.

Those formulations of movement primitives have found a large number of applications. They are used, for instance, to model Human-Robot Interaction. The Interaction Primitives (IPs), proposed by Ben Amor at al. [39], are based on DMPs. The Interaction Probabilistic Movement Primitives, proposed by Maeda et al. [36], and the Mixture of Interaction Primitives, proposed by Ewerton et al. [37], are based on ProMPs. All those interaction models are time-dependent. In their current form, those models require the time-alignment of the demonstrations. In [39], Dynamic Time Warping [40] is used to time-align the demonstrations. In [36], a local optimization method is proposed to this end. This same method is also used in [37]. However, as will be discussed in Section 2.4.1.2, those procedures do not allow for reacting to actions executed at different speeds.

Time-independent methods to learn trajectories by imitation have also been proposed. For example, Calinon et al. [41] propose an approach based on Hidden Markov Models (HMMs) and Gaussian mixture regression (GMR) to learn and reproduce gestures by imitation. Each hidden state corresponds to a Gaussian over positions and velocities, locally encoding variation and correlation. ProMPs, however, offer very useful properties for learning motor skills and approaching the problem of Human-Robot Interaction, such as the parameterization of trajectories with a relatively small set of weights and the global encoding of variation and correlation. Therefore, ProMPs have been chosen in our work.

A number of other methods deal with the problem of phase estimation or time-alignment. Englert et al. [42] present a method to adapt the trajectory and the phase of a movement to changes in the environment, such as changes in the position of the goal or in the position of obstacles. Vuga et al. [43] present a modified form of DMPs where the rate of phase change is related to the speed of movement. They use Reinforcement Learning and Iterative Learning Control (ILC) to speed up the execution of a movement as much as possible without violating some given constraints. For example, they speed up the movements of a robot carrying a glass full of liquid without spilling the liquid. Coates et al. [44] learn how to follow a desired trajectory from multiple sub-optimal expert's demonstrations. They apply their algorithm to the problem of autonomous helicopter flight. They use Dynamic Time Warping to find the relation between the time steps of the demonstrations and the time steps of the desired trajectory. Similarly, van den Berg et al. [45] use Dynamic Time Warping to learn the time mapping between demonstrated trajectories and a reference trajectory. With their approach, robots are able to perform surgical tasks with superhuman performance.

Differently from the cited works, the work on online phase estimation presented in this chapter aims at inferring the phase from a partial observation of a trajectory. The position of the goal is not known a priori and there is no reference trajectory.

Meier et al. [46] propose a technique to perform movement recognition, prediction and segmentation. This technique assumes the existence of a library of DMPs with specific weights and uses a reformulation of DMPs as linear dynamical systems. Given a partial observation of a trajectory, an EM algorithm is able to estimate the most probable primitive corresponding to this trajectory, the duration and the goal of this primitive. Their work is related to ours, especially since they also use an EM algorithm that can also estimate the completion of trajectories. In our work, an EM algorithm is used to infer a different set of parameters of movement primitives, specifically, the weights and the phase parameters of ProMPs. While their technique is especially suitable for movement segmentation, ours is able to model the correlation between positions at different time steps and the correlation between different joints, which is suitable for Human-Robot Interaction applications, for instance.

2.3 EM Algorithm to Learn ProMPs

This section explains an EM algorithm to determine the vectors of weights that compactly represent the training movements of a ProMP and a probability distribution over those weights as well as the phase parameters for each training movement and a probability distribution over those phase parameters. First, the algorithm is introduced for the case in which the phase function can be defined by one single parameter. Afterward, an extension of this method for the case in which the phase function depends on multiple parameters is presented.

2.3.1 Algorithm with a Single Phase Parameter

Assume there are a number K of training movements in the form of trajectories. Each training movement can be compactly represented by a vector $\boldsymbol{w}$ of weights for basis functions[1] and, for now, by a single non-negative phase parameter α. The phase function $z(t)$ is defined as $z(t) = \alpha t$ and assumes values between 0 and Z. The higher the value of α, the faster the phase goes from 0 to Z and the faster the movement gets executed.
The phase parameter α_i of a trajectory indexed by i is given by

$$\alpha_i = \frac{Z}{T_i}, \tag{2.1}$$

where T_i is the duration of the training trajectory i.
Assuming $p_\theta(\boldsymbol{w})$ is a multivariate Gaussian defined by a set of parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$, i.e. its mean and covariance, one of the objectives of the algorithm is to determine $\boldsymbol{\theta}$. The training trajectories may have missing values, as it would be the case for example due to occlusion or lack of sensor coverage. The observed part of trajectory i is represented by D_i. Finding the parameters $\boldsymbol{\theta}$ of $p_\theta(\boldsymbol{w})$ can be formulated as an Expectation-Maximization [47] problem, where $\boldsymbol{w}$ is a hidden, vector-valued variable.
The joint probability of the observations[2]

$$\prod_i p_{\boldsymbol{\theta}}(D_i|\alpha_i) = \prod_i \int p(D_i|\boldsymbol{w}, \alpha_i)\, p_{\boldsymbol{\theta}}(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}, \tag{2.2}$$

with $p(D_i|\boldsymbol{w}, \alpha_i) = \mathcal{N}\left(D_i; \boldsymbol{\Psi}_i \boldsymbol{w}, \sigma^2 \boldsymbol{I}\right)$, must be maximized with respect to $\boldsymbol{\theta}$. The term $\sigma^2 \boldsymbol{I}$ is a covariance matrix modeling a uniform observation noise.
Assuming that an observation D_i comprises the positions q_t from time step $t = 1$ to time step $t = m$, where $m \leq T_i$, this observation can be represented by

$$D_i = [q_1, q_2, \cdots, q_m]^\top \approx \boldsymbol{\Psi}_i \boldsymbol{w}_i, \tag{2.3}$$

where $\boldsymbol{w}_i$ is the unobserved vector of weights that compactly represents the training trajectory i. We assume a number N of basis functions, i.e. the weight vectors are given by $\boldsymbol{w}_i = [w_{i1}, w_{i2}, \cdots, w_{iN}]^\top$.
The matrix $\boldsymbol{\Psi}_i$ has each basis function ψ_n evaluated at the phase value corresponding to the observed time steps t. This matrix can be written as

$$\boldsymbol{\Psi}_i = \begin{bmatrix} \psi_1(z_i(1)) & \psi_2(z_i(1)) & \cdots & \psi_N(z_i(1)) \\ \psi_1(z_i(2)) & \psi_2(z_i(2)) & \cdots & \psi_N(z_i(2)) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(z_i(m)) & \psi_2(z_i(m)) & \cdots & \psi_N(z_i(m)) \end{bmatrix}, \tag{2.4}$$

where $z_i(t) = \alpha_i t$.
Note that the observations D_i do not need to be comprised of positions at successive time steps. There may be gaps between the observed parts as well.
As the parameter vector $\boldsymbol{w}_i$ for each demonstration D_i is unobserved, an EM algorithm is used to maximize the likelihood of the demonstrations. First, $\boldsymbol{\theta}$ is initialized with a rough estimate $\boldsymbol{\theta}_0 = \{\boldsymbol{\mu}_{w0}, \boldsymbol{\Sigma}_{w0}\}$. Next, the algorithm performs the Expectation step (E step), by estimating for each observation D_i the posterior $p_{\theta_0}(\boldsymbol{w}|D_i, \alpha_i) \propto p(D_i|\boldsymbol{w}, \alpha_i) p_{\theta_0}(\boldsymbol{w})$, which is a Gaussian with mean $\boldsymbol{\mu}_{wi}$ and covariance $\boldsymbol{\Sigma}_{wi}$. The posterior for each observation D_i is necessary to define the complete data log-likelihood $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}})$,

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}}) = \sum_i \mathbb{E}_{\boldsymbol{\theta}_{\text{old}}} [\log p_{\boldsymbol{\theta}}(D_i, \boldsymbol{w}, \alpha_i) | D = D_i], \tag{2.5}$$

which is maximized with respect to $\boldsymbol{\theta}$ in the Maximization step (M step), i.e.

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}}). \tag{2.6}$$

[1] We use normalized Gaussian basis functions.
[2] Assuming $\boldsymbol{w}$ and α independent variables.

The algorithm keeps iterating over the E step and the M step until $\prod_i p_{\boldsymbol{\theta}}(D_i|\alpha_i)$ converges. The parameters $\boldsymbol{\mu_{wi}}$, $\boldsymbol{\Sigma}_{wi}$ and Equation (2.6) have closed-formed solutions with
E step

$$\boldsymbol{\Sigma}_{wi} = \left(\sigma^{-2}\boldsymbol{\Psi}_i^\top \boldsymbol{\Psi}_i + \boldsymbol{\Sigma}_{w_{\text{old}}}^{-1}\right)^{-1}, \tag{2.7}$$

$$\boldsymbol{\mu}_{wi} = \boldsymbol{\Sigma}_{wi}\left(\sigma^{-2}\boldsymbol{\Psi}_i^\top D_i + \boldsymbol{\Sigma}_{w_{\text{old}}}^{-1}\boldsymbol{\mu}_{w_{\text{old}}}\right) \tag{2.8}$$

and **M step**

$$\boldsymbol{\mu}_w^* = \frac{\sum_i \boldsymbol{\mu}_{wi}}{K}, \tag{2.9}$$

$$\boldsymbol{\Sigma}_w^* = \frac{\left(\boldsymbol{E}^\top \boldsymbol{E} + \sum_i \boldsymbol{\Sigma}_{w_i}\right)}{K}, \tag{2.10}$$

where

$$\boldsymbol{\mu}_{wi} = \left[\mu_{w_{i1}}, \mu_{w_{i2}}, \cdots, \mu_{w_{iN}}\right]^\top,$$
$$\boldsymbol{\mu}_w^* = \left[\mu_{w_1}^*, \mu_{w_2}^*, \cdots, \mu_{w_N}^*\right]^\top,$$
$$\boldsymbol{e}_i = \boldsymbol{\mu}_{wi} - \boldsymbol{\mu}_w^*, \quad \boldsymbol{E} = \left[\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_K\right]^\top.$$

Note that this approach to learn ProMPs incorporates all available observations when estimating the mean $\boldsymbol{\mu}_{wi}$ and the covariance $\boldsymbol{\Sigma}_{wi}$, which define the probability distribution over the weights for each training trajectory i. Equations (2.7) and (2.8) involve the terms $\boldsymbol{\mu}_{w_{\text{old}}}$ and $\boldsymbol{\Sigma}_{w_{\text{old}}}$, which represent the mean and the covariance from the previous EM iteration of the Gaussian over the weights of all the training trajectories.

2.3.2 Algorithm with Multiple Phase Parameters

A set of movement demonstrations may present local variabilities in the speed of execution. For example, consider a data set in which the beginning of the movements has approximately the same speed of execution, but the end differs considerably in speed. A phase function with one single parameter cannot account for this kind of variability because changing its phase parameter would result in accelerating or decelerating the whole movement.
In order to learn movement primitives that model local variabilities in the speed of execution, the rate of change $\dot{z}(t)$ of the phase function with respect to time t can be defined as a weighted sum of Gaussian basis functions,

$$\dot{z}(t) = \boldsymbol{\phi}(t)^\top \boldsymbol{\alpha}, \tag{2.11}$$

where $\boldsymbol{\phi}(t)$ is a vector of normalized Gaussian basis functions evaluated at time step t and $\boldsymbol{\alpha}$ is a vector of phase parameters, which are the weights for the basis functions.
The phase function can be obtained by first defining its value at time step $t = 0$ and then using Euler Integration until a time limit t_{max} is achieved,

$$z(0) = 0, \quad z(t+1) = z(t) + \Delta t \dot{z}(t).$$

However, once $z(t) = Z$, it no longer increases, remaining with the value Z until $t = t_{max}$. Movements may have any duration that does not surpass the time limit t_{max}.
When learning ProMPs with multiple phase parameters, the estimation of the phase parameters is incorporated into the EM algorithm. This time, $\boldsymbol{w}$ and $\boldsymbol{\alpha}$ are both hidden, vector-valued variables. The set of parameters $\boldsymbol{\theta}$ is initialized with a rough estimate $\boldsymbol{\theta}_0 = \left\{\boldsymbol{\mu}_{w0}, \boldsymbol{\Sigma}_{w0}, \boldsymbol{\mu}_{\alpha 0}, \boldsymbol{\Sigma}_{\alpha 0}\right\}$. At the beginning of each iteration of the algorithm, a number S of vectors $\boldsymbol{\alpha}_j$ are sampled from $\mathcal{N}\left(\boldsymbol{\alpha}; \boldsymbol{\mu}_{\alpha_{\text{old}}}, \boldsymbol{\Sigma}_{\alpha_{\text{old}}}\right)$, where $\boldsymbol{\mu}_{\alpha_{\text{old}}}$ and $\boldsymbol{\Sigma}_{\alpha_{\text{old}}}$ are parameters determined by the previous iteration. The algorithm optimizes the parameter vector $\boldsymbol{\theta} = \left\{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha\right\}$ to maximize $\prod_{i,j} p_{\boldsymbol{\theta}}\left(D_i|\boldsymbol{\alpha}_j\right)$. This operation is performed by executing iteratively an EM to determine $\left\{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right\}$ and an EM to determine $\left\{\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha\right\}$ with
E step for $\boldsymbol{w}$

$$\boldsymbol{\Sigma}_{wij} = \left(\sigma^{-2}\boldsymbol{\Psi}_{ij}^\top \boldsymbol{\Psi}_{ij} + \boldsymbol{\Sigma}_{w_{\text{old}}}^{-1}\right)^{-1}, \tag{2.12}$$

$$\boldsymbol{\mu}_{wij} = \boldsymbol{\Sigma}_{wij}\left(\sigma^{-2}\boldsymbol{\Psi}_{ij}^\top D_{ij} + \boldsymbol{\Sigma}_{w_{\text{old}}}^{-1}\boldsymbol{\mu}_{w_{\text{old}}}\right) \tag{2.13}$$

and **M step for $\boldsymbol{w}$**

$$\boldsymbol{\mu}_w^* = \frac{\sum_{i,j} p\left(\boldsymbol{\alpha}_j|D_i\right)\boldsymbol{\mu}_{wij}}{\sum_{i,j} p\left(\boldsymbol{\alpha}_j|D_i\right)}, \tag{2.14}$$

$$\boldsymbol{\Sigma}_w^* = \frac{\left(\boldsymbol{E}_d^\top \boldsymbol{E} + \sum_{i,j} p\left(\boldsymbol{\alpha}_j|D_i\right)\boldsymbol{\Sigma}_{w_{ij}}\right)}{\sum_{i,j} p\left(\boldsymbol{\alpha}_j|D_i\right)}, \tag{2.15}$$

where

$$\begin{aligned}
&\boldsymbol{\mu}_{wij} = \left[\mu_{w_{ij1}}, \mu_{w_{ij2}}, \cdots, \mu_{w_{ijN}}\right]^\top,\\
&\boldsymbol{\mu}_w^* = \left[\mu_{w_1}^*, \mu_{w_2}^*, \cdots, \mu_{w_N}^*\right]^\top,\\
&\boldsymbol{e}_{ij} = \boldsymbol{\mu}_{wij} - \boldsymbol{\mu}_w^*, \quad \boldsymbol{E} = [\boldsymbol{e}_{11}, \boldsymbol{e}_{12}, \cdots, \boldsymbol{e}_{KS}]^\top,\\
&\boldsymbol{e}_{dij} = p\left(\boldsymbol{\alpha}_j|\boldsymbol{D}_i\right)\left(\boldsymbol{\mu}_{wij} - \boldsymbol{\mu}_w^*\right),\\
&\boldsymbol{E}_d = [\boldsymbol{e}_{d11}, \boldsymbol{e}_{d12}, \cdots, \boldsymbol{e}_{dKS}]^\top.
\end{aligned}$$

Having executed the EM for $\boldsymbol{w}$, the algorithm recomputes the terms $p\left(\boldsymbol{\alpha}_j|D_i\right)$ according to $\left\{\boldsymbol{\mu}_w^*, \boldsymbol{\Sigma}_w^*\right\}$ and executes the EM for $\boldsymbol{\alpha}$ with

E step for $\boldsymbol{\alpha}$

$$\mathbb{E}[\boldsymbol{\alpha}|D_i] = \frac{\sum_j p\left(\boldsymbol{\alpha}_j|D_i\right)\boldsymbol{\alpha}_j}{\sum_j p\left(\boldsymbol{\alpha}_j|D_i\right)}, \tag{2.16}$$

and **M step for $\boldsymbol{\alpha}$**

$$\boldsymbol{\mu}_\alpha^* = \frac{\sum_i \mathbb{E}[\boldsymbol{\alpha}|D_i]}{K}, \tag{2.17}$$

$$\boldsymbol{\Sigma}_\alpha^* = \frac{\sum_{i,j} p\left(\boldsymbol{\alpha}_j|D_i\right)\left(\boldsymbol{\alpha}_j - \boldsymbol{\mu}_\alpha^*\right)\left(\boldsymbol{\alpha}_j - \boldsymbol{\mu}_\alpha^*\right)^\top}{\sum_{i,j} p\left(\boldsymbol{\alpha}_j|D_i\right)}. \tag{2.18}$$

The mean $\boldsymbol{\mu}_\alpha^*$ and the covariance $\boldsymbol{\Sigma}_\alpha^*$ become the mean $\boldsymbol{\mu}_{\alpha_{\text{old}}}$ and the covariance $\boldsymbol{\Sigma}_{\alpha_{\text{old}}}$ of the Gaussian probability distribution from which the vectors $\boldsymbol{\alpha}_j$ are sampled at the beginning of the next iteration of the algorithm. The iterations continue until $\prod_{i,j} p_{\boldsymbol{\theta}}\left(D_i|\boldsymbol{\alpha}_j\right)$ converges.

2.3.3 Online Phase Estimation and Movement Prediction

Given a set D_i of observed positions at specific time steps of a movement i being observed[3], the phase associated with this movement can be online estimated and the unobserved part of this movement can subsequently be predicted, as long as its trajectory and phase fit into the probability distributions learned in the training phase.

In order to perform this prediction, a number of vectors $\boldsymbol{\alpha}_j$ are sampled from $\mathcal{N}\left(\boldsymbol{\alpha}; \boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha\right)$ resulting from the training phase performed with the algorithm explained in Section 2.3.2. The probability of $\boldsymbol{\alpha}_j$ given observation D_i is given by[4]

$$p\left(\boldsymbol{\alpha}_j|D_i\right) \propto p\left(D_i|\boldsymbol{\alpha}_j\right), \tag{2.19}$$

where[5]

$$p\left(D_i|\boldsymbol{\alpha}_j\right) = \int p\left(D_i|\boldsymbol{w}, \boldsymbol{\alpha}_j\right) p\left(\boldsymbol{w}\right) \mathrm{d}\boldsymbol{w}. \tag{2.20}$$

[3] This movement does not need to be part of the training data. In fact, for any practical application, this movement most likely does not match exactly any of the demonstrations.

[4] Note that $p(\boldsymbol{\alpha}_j)$ is not part of the expression, since the vectors $\boldsymbol{\alpha}_j$ are being sampled.

[5] Assuming $\boldsymbol{w}$ and $\boldsymbol{\alpha}$ independent variables.

Equation 2.20 can be solved in closed form, yielding $p\left(D_i|\boldsymbol{\alpha}_j\right) = \mathcal{N}\left(D_i; \boldsymbol{\mu}_{Dij}, \boldsymbol{\Sigma}_{Dij}\right)$ with

$$\boldsymbol{\mu}_{Dij} = \boldsymbol{\Psi}_{ij}\boldsymbol{\mu}_{wij}, \tag{2.21}$$

$$\boldsymbol{\Sigma}_{Dij} = \sigma^2\boldsymbol{I} + \boldsymbol{\Psi}_{ij}\boldsymbol{\Sigma}_{wij}\boldsymbol{\Psi}_{ij}^\top. \tag{2.22}$$

The terms $\boldsymbol{\mu}_{wij}$ and $\boldsymbol{\Sigma}_{wij}$ can be computed using (2.13) and (2.12), respectively.
Finally, the mean $\boldsymbol{\mu}_\tau$ and covariance $\boldsymbol{\Sigma}_\tau$ of the predicted trajectory τ for the whole duration T_i of the movement i can be computed with

$$\boldsymbol{\mu}_\tau = \frac{\sum_j p\left(\boldsymbol{\alpha}_j|D_i\right)\boldsymbol{\mu}_{\tau j}}{\sum_j p\left(\boldsymbol{\alpha}_j|D_i\right)}, \tag{2.23}$$

$$\boldsymbol{\Sigma}_\tau = \frac{\sum_j p\left(\boldsymbol{\alpha}_j|D_i\right)\left(\boldsymbol{\Sigma}_{\tau j} + \boldsymbol{\mu}_{\tau j}\boldsymbol{\mu}_{\tau j}^\top - \boldsymbol{\mu}_\tau\boldsymbol{\mu}_\tau^\top\right)}{\sum_j p\left(\boldsymbol{\alpha}_j|D_i\right)}, \tag{2.24}$$

where

$$\boldsymbol{\mu}_{\tau j} = \boldsymbol{\Psi}_j\boldsymbol{\mu}_{wij}, \tag{2.25}$$

$$\boldsymbol{\Sigma}_{\tau j} = \sigma^2\boldsymbol{I} + \boldsymbol{\Psi}_j\boldsymbol{\Sigma}_{wij}\boldsymbol{\Psi}_j^\top. \tag{2.26}$$

While $\boldsymbol{\Psi}_{ij}$ is defined only at the time steps of observed positions, $\boldsymbol{\Psi}_j$ is defined over the whole duration T_i of the movement according to $\boldsymbol{\alpha}_j$.
This prediction is thus a weighted average of the predictions with all sampled vectors $\boldsymbol{\alpha}_j$. Equations (2.23) and (2.24) can be derived by minimizing the Kullback-Leibler divergence [47] between a mixture of Gaussians with parameters $\left\{p\left(\boldsymbol{\alpha}_j|D_i\right), \boldsymbol{\mu}_{\tau j}, \boldsymbol{\Sigma}_{\tau j}\right\}$ and a single Gaussian with parameters $\left\{\boldsymbol{\mu}_\tau, \boldsymbol{\Sigma}_\tau\right\}$.

2.4 Experiments

This section presents a number of experiments that evaluate some of the applications of our algorithm described in Section 2.3. Those applications are online phase estimation with subsequent movement prediction, learning from incomplete data and inferring distributions over trajectories associated with phase functions defined by multiple parameters.

2.4.1 Experiments on Online Phase Estimation and Movement Prediction

In this first set of experiments, we evaluate online phase estimation and movement prediction assuming all movements can be associated with a phase function defined by one single phase parameter. Each movement has, however, its own phase parameter α. This evaluation is performed in two scenarios: with artificially generated trajectories of the letter “a” and in a Human-Robot Interaction scenario, in which a robot receives a cup from a human.

2.4.1.1 Using Artificially Generated “a” Trajectories

Consider a data set comprising several (x, y) trajectories of the letter “a”, differing in shape, scale and duration. Fig. 2.2 exemplifies such a data set, showing different letters “a” and their corresponding x trajectories. Given a partial observation of a test trajectory, our objective is to predict the unobserved part. The prediction should be as close as possible to the ground truth. In the experiments presented in this section, the training data set was comprised only of whole trajectories, without any gaps.
The training and test data were constructed by sampling weights $\boldsymbol{w}$ for a set of 20 Gaussian basis functions and phase parameters α from Gaussian distributions. Forty trajectories were generated with the sampled values, using (2.3). Twenty of them were selected at random as training and the other twenty as test data. This data set fulfills

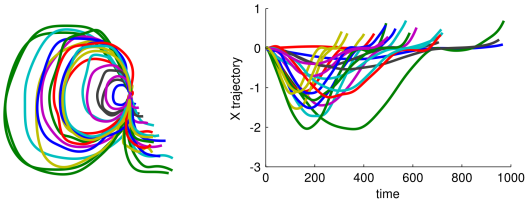


Figure 2.2.: Data set consisting of different trajectories of the letter "a". The trajectories differ in shape, scale and duration.

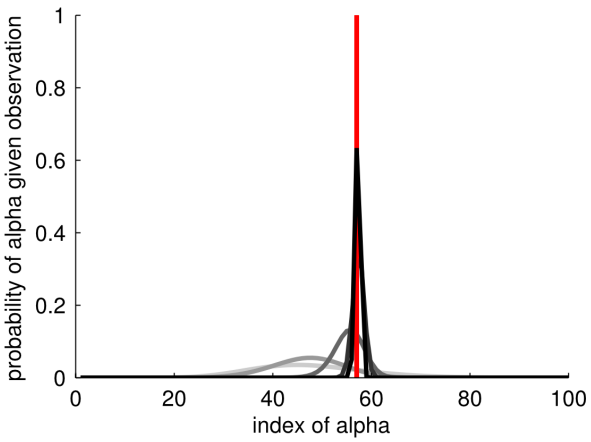


Figure 2.3.: Phase estimation after observing 5, 10, 20, 40 and 60 percent of a test trajectory. The darker the curve representing the probability density function, the larger the observed part. The vertical red line corresponds to the ground truth phase parameter.

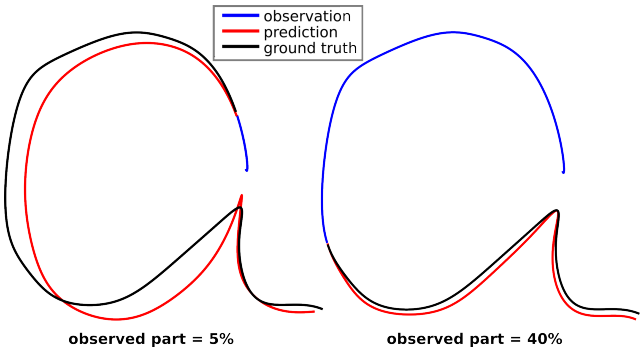


Figure 2.4.: Predictions after observing 5 and 40 percent of a test trajectory.

perfectly the assumptions that both the weights $\boldsymbol{w}$ of the basis functions and the phase parameters α are normally distributed. For this reason, it was chosen as a starting point to test the method proposed in Section 2.3.3 to estimate the phase online and predict the unobserved part of a movement.

In the training phase, the weights $\boldsymbol{w}$ for each training trajectory were computed with linear regression and the phase variables α were computed with Equation (2.1). Then, Gaussian probability distributions $p(\boldsymbol{w})$ and $p(\alpha)$ over the computed values were defined.

In the test phase, 50 values for the α parameter were sampled from the prior probability distribution $p(\alpha)$. For testing, we provided for each test trajectory observations with an increasing number of time steps. After each new observed position, the remaining test trajectory was predicted using the method described in Section 2.3.3. Fig. 2.3 shows the probability density function $p(\alpha|D)$ over the sampled values for α after observing 5%, 10%, 20%, 40% and 60% of a test trajectory. The vertical red line corresponds to the ground truth phase parameter. In general, the larger the observed part, the better the estimation of α and the better the prediction. Fig. 2.4 shows the prediction after observing 5% and 40% of a test trajectory in comparison to the ground truth.

The accuracy of the predictions was evaluated by computing the root-mean-square (RMS) error between prediction and ground truth. The RMS error was computed for each of the twenty test trajectories every time the number of time steps of the observed part increased 1% in relation to the total number of time steps of the trajectory. Fig. 2.5 shows the mean and the variance of the RMS error as the observed part increases.

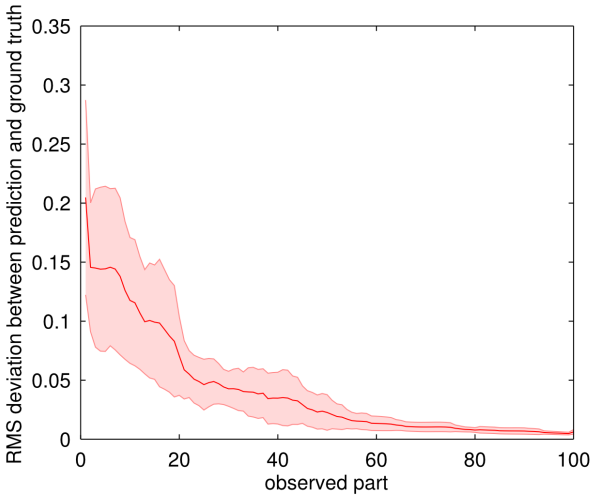


Figure 2.5.: Observed part versus RMS error with mean and standard deviation.

2.4.1.2 Human-Robot Interaction Experiment

We evaluated the same algorithm on a human-robot collaborative task of an object handover (see Fig. 2.6).

During training, the human moved with different speeds in the direction of the robot but also varying the position at which he handed over the cup, while the robot was moved by kinesthetic teaching to receive the cup at the correct

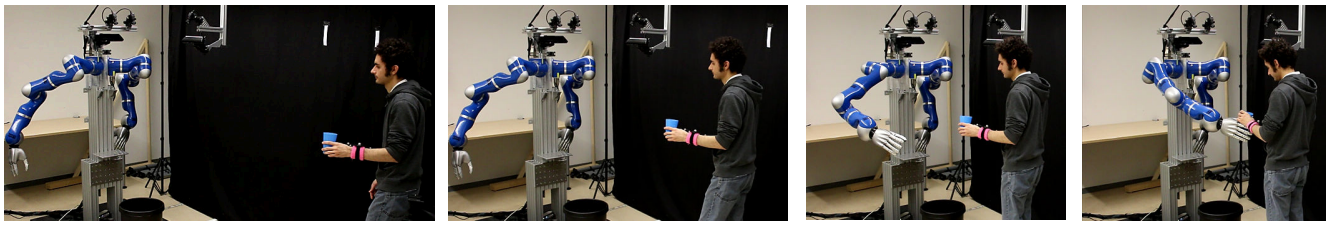


Figure 2.6.: A sequence of snapshots of a cup handover interaction between a human and a robot. The robot infers online the phase of the human movement and computes the expected reaction to this movement, according to the training.

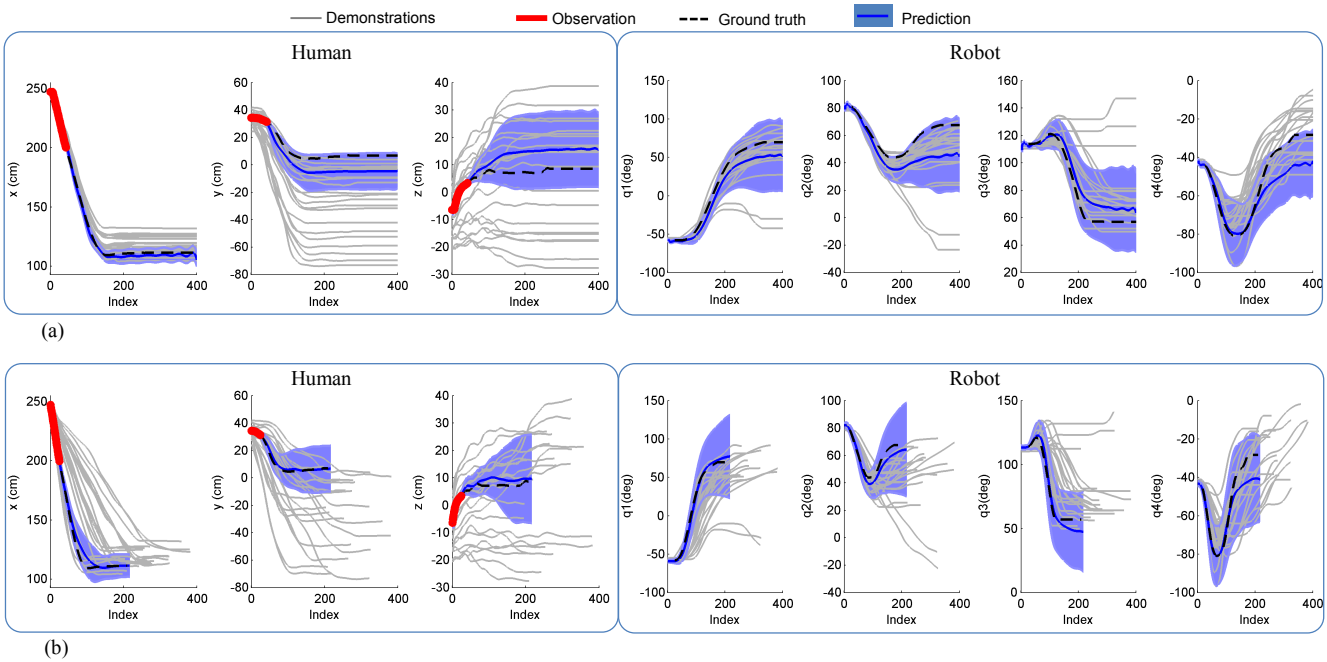


Figure 2.7.: One of the test cases where the robot reaction is computed based on a partial observation of the human movement. The (x, y, z) coordinates of the human wrist and four joint angles of the 7-DoF robotic arm are shown. (a) Conditioning after time-alignment with the method proposed in [36], which does not allow for online conditioning. (b) Conditioning online, without time-alignment, using the method proposed in this chapter.

position. The trajectories of the human were about 150 cm long. Each of those trajectories was a sequence of (x, y, z) coordinates of the human's left wrist, which had markers attached to it detectable by a motion capture system.

During test, the human was observed only during the first 50 cm of his trajectory. Then, 15 values for the phase parameter α were sampled from the $p(\alpha)$ learned in the training phase. Subsequently, the rest of the movement of the human and the expected reaction of the robot[6] were computed online using the method presented in Section 2.3.3.

Fig. 2.7 shows one of the test cases from a leave-one-out cross-validation (LOOCV) procedure run over a data set of recorded trajectories (22 in total). Fig. 2.7(a) shows the result of conditioning the learned probability distribution over trajectories on the observed sequence of positions of the human with the method proposed in [36]. The method proposed in that work performs first the time-alignment of the recorded trajectories using a local optimization algorithm and then performs the conditioning. That method cannot condition online on the beginning of the human's movement, since it is not possible to time-align his trajectory before it has been completed. Fig. 2.7(b) shows the result of our method, in which the training movements, in gray, preserve their original speed profile. Using the same observations as in the previous case, the robot's joint trajectories could still be predicted with similar accuracy, but now also with variable phases, inferred from the human movement.

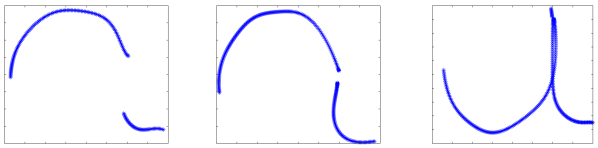


Figure 2.8.: Training trajectories with missing data points.

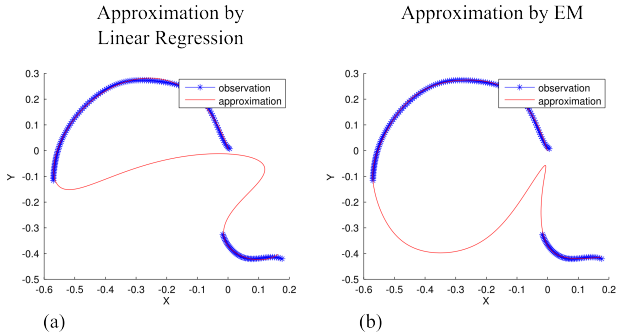


Figure 2.9.: Approximation to a training trajectory by linear regression versus approximation by EM.

2.4.2 Experiments on Dealing with Missing Data

This section presents experiments that show the applicability of the EM algorithm proposed in Section 2.3 to learn ProMPs in the face of training trajectories with missing data points.

The training and test trajectories were artificially generated letters "a" as in Section 2.4.1.1. This time, however, random sequences of 180 time steps were removed from the training trajectories. Since the shortest training trajectory comprised 355 time steps and the longest, 1129, the missing parts corresponded approximately to between 16% and 51% of the entire trajectories. Fig. 2.8 shows three of those training trajectories.

In a first experiment, linear regression was used to learn the weight vectors $\boldsymbol{w}$ for each of those training trajectories and a Gaussian distribution was fitted to the resulting set of weight vectors. Fig. 2.9(a) shows the approximation generated by linear regression to one of the training trajectories. In a second experiment, the EM algorithm proposed in Section 2.3.1 was used instead of linear regression to learn the weights $\boldsymbol{w}$. Fig. 2.9(b) shows the approximation generated by the EM algorithm to the same training trajectory as in Fig. 2.9(a).

The approximation generated by the EM algorithm resembles a letter "a", while the one generated by the linear regression method does not. The reason is that, while learning the weights for one training trajectory, the linear regression method only takes into consideration the observed positions of this trajectory, not using any information from other training trajectories, which may have observed positions complementing the observations currently under consideration. The EM algorithm, on the other hand, takes into consideration all training trajectories while computing the mean and the covariance that define a probability distribution over the weights $\boldsymbol{w}$ for each trajectory i with (2.7) and (2.8). Note that those equations involve the terms $\boldsymbol{\mu}_{w_{\text{old}}}$ and $\boldsymbol{\Sigma}_{w_{\text{old}}}$, which represent the mean and the covariance from the previous EM iteration of the Gaussian over the weights of all the training trajectories.

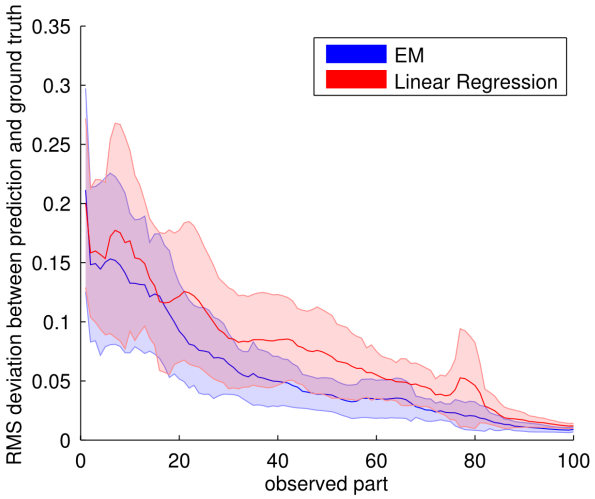


Figure 2.10.: Comparison between error of the predictions after training with EM and error of the predictions after training with linear regression, for the case in which the training trajectories have missing data points.

After training a ProMP with the linear regression method and another ProMP with the EM method, both models were used to estimate the phase and predict the unobserved part of test trajectories. Fig. 2.10 shows the mean and the standard deviation of the RMS error as the observed part of the test trajectories increases. As the observed part gets larger, the RMS gets in general lower for both methods because the prior probability distribution over the weights $\boldsymbol{w}$ gets less and less relevant in the face of new observations. However, the EM approach reaches lower values for the RMS sooner than the linear regression method.

[6] For more details on how to apply the ProMP framework to interaction scenarios, the interested reader is referred to [36].

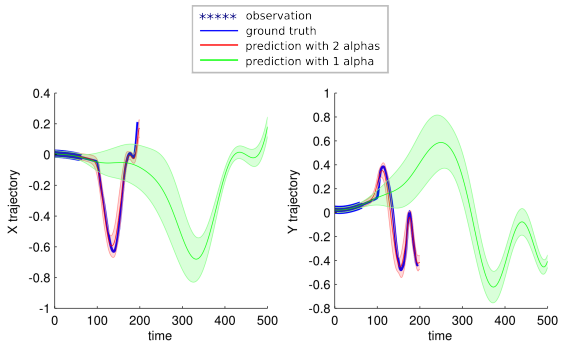


Figure 2.11.: Comparison between prediction ($\mu \pm 2\sigma$) of the model with two phase parameters (red) and prediction of the model with only one phase parameter (green), having observed 30% of a test trajectory. The training and test trajectories were in this case artificially generated letters "a".

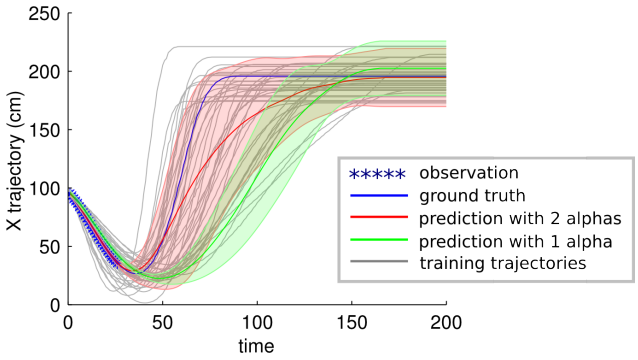


Figure 2.12.: Comparison between prediction ($\mu \pm 2\sigma$) of the model with two phase parameters (red) and prediction of the model with only one phase parameter (green), having observed part of a test trajectory before the minimum value of x had been reached. The training and test trajectories were in this case golf putting trajectories demonstrated by a human.

2.4.3 Experiments on Using Multiple Phase Parameters

In this section, we evaluate differences between models with only one phase parameter and models with multiple phase parameters. In the experiments here presented, whole trajectories, with no missing data, were used for training. Nevertheless, the data sets used here have speed profiles that make predicting unobserved parts more challenging than with the previously used data sets.

In a first set of experiments, training and test data comprised artificially generated "a" trajectories. This time, however, these trajectories were associated with phase functions with two phase parameters, i.e. these trajectories could for instance start slowly and end fast or start fast and end slowly. After training and having observed a part of a test trajectory, our objective was to predict the unobserved part. The prediction should be as close as possible to the ground truth. Fig. 2.11 shows the prediction produced by the model with two phase parameters, in red, and the prediction produced by the model with only one phase parameter, in green, after observing 30% of a test trajectory. The model with two phase parameters produced a much better prediction because it accounts for local changes in the speed of execution.

We performed the same experiment also using a data set of golf putting movements executed by a human. Fig. 2.12 shows the prediction produced by the model with two phase parameters, in red, and the prediction produced by the model with only one phase parameter, in green, after observing a sequence of positions at the beginning of a test trajectory. This sequence of observations comprised positions before the x degree of freedom (DoF) had reached its minimum value. The x DoF is the one with the largest range of values, from left to right in Fig. 2.1(b). While both predictions are considerably far away from the ground truth, the prediction from the model with two phase parameters has a covariance that better represents the training trajectories, what can be observed in Fig. 2.12 by the fact that the red shade covers more of the training trajectories than the green shade. Moreover, the prediction of the model with two phase parameters had an average RMS error of approximately 18.93 cm with a standard deviation of 8.23 cm, while the model with only one phase parameter had an average RMS error of approximately 24.69 cm with a standard deviation of 13.52 cm.

2.5 Epilogue

This chapter presented an *Expectation-Maximization* algorithm to learn motor skills in the form of *Probabilistic Movement Primitives*. The presented algorithm allows for learning from trajectories with missing data and accounts for the spatial-temporal variability of the demonstrations. Experiments demonstrated the applicability of this algorithm to movement prediction and to Human-Robot Interaction scenarios in which the robot must react to human movements executed at different speeds.

Possible extensions of this work involve using Gaussian mixture models to account for nonlinear correlations between joints or interacting agents and to learn multiple tasks as in [37]. Applications to real data of models with multiple phase parameters deserve further investigation, as well as different phase function formulations with different numbers of parameters.

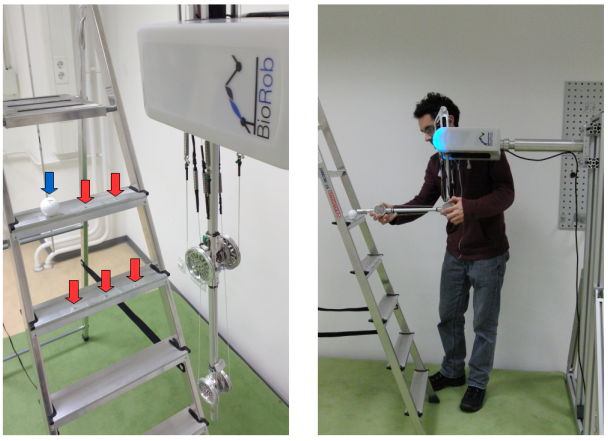
3 Incremental Imitation Learning of Context-Dependent Motor Skills

Teaching motor skills to robots through human demonstrations, an approach called “imitation learning”, is an alternative to hand coding each new robot behavior. Imitation learning is relatively cheap in terms of time and labor and is a promising route to give robots the necessary functionalities for widespread use in households, stores, hospitals, etc. However, current imitation learning techniques struggle with a number of challenges that prevent their wide usability. For instance, robots might not be able to accurately reproduce every human demonstration and it is not always clear how robots should generalize a movement to new contexts. This chapter addresses those challenges by presenting a method to incrementally teach context-dependent motor skills to robots. The human demonstrates trajectories for different contexts by moving the links of the robot and partially or fully refines those trajectories by disturbing the movements of the robot while it executes the behavior it has learned so far. A joint probability distribution over trajectories and contexts can then be built based on those demonstrations and refinements. Given a new context, the robot computes the most probable trajectory, which can also be refined by the human. The joint probability distribution is incrementally updated with the refined trajectories. We have evaluated our method with experiments in which an elastically actuated robot arm with four degrees of freedom learns how to reach a ball at different positions.

3.1 Prologue

Since the beginning of the 1980s, a large amount of research has been done to make robots capable of learning motor skills by imitating human demonstrations. This strategy, called “imitation learning” [1], might become a viable route to quickly train general-purpose robots to perform new tasks on demand in environments such as households, stores, hospitals, etc.

However, a number of challenges hinder the widespread application of imitation learning in those environments. For example, the robot might not be able to accurately reproduce some of the movements demonstrated by the human and the robot should be able to generalize the motor skills it has learned so far to different contexts. In those cases, it may be helpful to structure the imitation learning as an incremental process. In this process, the human could incrementally refine the robot trajectories in order to make it accomplish a certain task or incrementally correct trajectories inferred by the robot in the face of new contexts. In this way, the human would not have to demonstrate a movement all over again when the robot had made a mistake. Instead, the human could just give small incremental feedbacks that would be interpreted by the robot as necessary changes to its movements and to the relation between movements and contexts.



(a) Experimental setup (b) Kinesthetic teaching

Figure 3.1.: Images of an experiment involving the BioRob, a 4-DoF elastically actuated robot arm. The objective in this experiment is to teach the robot how to reach a ball at each of the positions pointed by the arrows. **(a)** The blue arrow points to the current position of the ball, while the red arrows point to other possible ball positions. **(b)** The human demonstrates through kinesthetic teaching how to reach the ball at a certain position.

Following this perspective, the main contribution of this chapter is an algorithm that allows humans to teach robots context-dependent motor skills through demonstrations in an incremental manner. Our demonstrations have been obtained through kinesthetic teaching, i.e. by letting the human grasp and move the links of the robot. The proposed algorithm allows the human to incrementally refine the movement of the robot, a desirable feature if the robot cannot imitate the original demonstration of the human or if the movement executed by the robot does not yet solve the task at hand.

In our method, a joint probability distribution over refined trajectories and context variables is built. Having built this joint distribution, the robot is able to infer from previous experiences what movement it should execute to accomplish

a certain task given a new context. The contexts used in our experiments have been in the form of via points. More generally, the context could for example be positions, orientations or other properties of objects in the workspace of the robot. The human can also incrementally correct the inferences of the robot by intervening in its movements. Each new refined trajectory and context is used to update the joint distribution.

The algorithm proposed in this work is fairly general and may be applied to teach different motor skills to robots. In our experiments, this algorithm has been applied in a 2D problem to make a particle pass through certain via points, with the BioRob [48], an elastically actuated robot arm with four degrees of freedom (DoFs), to make it reach a ball at different positions and in a minigolf-like task involving the same robot.

The remainder of this chapter is organized as follows: Section 3.2 presents related work. Section 3.3 introduces the proposed method for incremental imitation learning of context-dependent motor skills. Our method is explained starting with a procedure to learn trajectories for a single context. After that, the necessary extensions to deal with context-dependent motor skills are described. Section 3.4 presents our experiments. Finally, Section 3.5 summarizes the chapter and discusses ideas for future work.

3.2 Related Work

While in this work we have been specifically dealing with demonstrations through kinesthetic teaching, another form of imitation learning involves observation. In this case, the movements of a human are recorded by a camera or by a motion capture system. Those movements are mapped to the kinematics of the robot, which reproduces the demonstration or learns motion primitives from multiple demonstrations as in [49]. Usually, though, this mapping does not always produce the most preferable motions. For this reason, researchers have been investigating the idea of incremental imitation learning through kinesthetic teaching to refine the movements initially learned by the robot through observation.

Calinon and Billard [50] presented an approach based on Principal Component Analysis (PCA) and Gaussian Mixture Models (GMMs) to teach gestures to a humanoid robot. The gestures are demonstrated by a human in two ways: the human moves while sensors attached to his/her body record his/her movements or the human performs kinesthetic teaching by grasping and moving the arms of the robot. The robot learns new gestures from multiple human demonstrations and its movements can be incrementally refined by the human. When performing kinesthetic teaching, the human can decide which DoFs he/she wants to drive and which DoFs should be autonomously driven by the robot. The DoFs driven by the human are set in passive mode. In contrast, in our method, the human disturbs the movements of the robot without setting any DoFs in passive mode. In our work, the disturbances introduced by the human lead to changes in the behavior of the robot. The human does not need to demonstrate the whole movement of a DoF again in case it is not moving correctly yet. He/she just needs to apply incremental changes to this movement.

Lee and Ott [51] also presented a method to teach motor skills to a humanoid robot. In their method, first, the robot observes the movements of a human. Subsequently, a human may incrementally refine the movement of the robot through kinesthetic teaching. Their work uses the concept of a "motion refinement tube", which is a region around the nominal trajectory followed by the robot where the controller has low stiffness, allowing the human to refine the trajectory. Movements are represented in their method by Hidden Markov Models (HMMs). Changes introduced in a trajectory by the human translate into changes in the parameters of the HMM representing that trajectory. The updated HMM generates an updated trajectory, accounting for the refinements introduced by the human. Their approach offers desirable properties, such as the possibility of defining the "refinement tube" in such a way that changes in the trajectory of one joint do not result in accidental disturbances in the trajectories of other joints. In contrast to our work, their approach requires an accurate model of the robot dynamics in order to identify human intervention. Our approach allows robots to learn context-dependent motor skills in the absence of an accurate model of the robot dynamics at the expense of having the robot execute a movement with and without human intervention at each iteration of the algorithm.

Another approach to incremental imitation learning aims at eliciting human preferences from the feedbacks he/she gives to the robot. Jain et al. [52] have developed a framework that enables robots to learn grocery checkout tasks with the help of human feedback. In their work, it is assumed that the user has an unknown score function quantifying the quality of trajectories in different contexts. This score function is a weighted sum of predefined features describing: 1) interactions between objects, 2) robot arm configurations, 3) orientation and temporal behavior of the object being manipulated, 4) interactions between the object being manipulated and the environment. The robot learns the weights for each of those features from human feedback. The user gives feedback by re-ranking a set of trajectories proposed by the robot or by changing waypoints of trajectories, setting the robot in zero-force gravity-compensation mode. In our method, when giving physical feedback to the robot, the user does not change specific waypoints. Instead, the user disturbs the movement of the robot while it is moving. By doing so, the user can directly interfere in the speed profile of the trajectory of the robot, teaching it for example to hit a golf ball faster or slower. Our work also differs from the cited one by computing a probability distribution between trajectories and contexts. While the computation of the distribution requires modeling assumptions (Gaussians, GMMs, etc.), it does not require the design of a score function.

Akgun et al. [53] conducted a study evaluating kinesthetic teaching by demonstrating trajectories as well as kinesthetic teaching by demonstrating keyframes. They concluded that both trajectory and keyframes demonstration are viable methods to teach motor skills to humanoid robots and have their specific advantages. When keyframes were demonstrated, the trajectories were generated by splines connecting the keyframes. They presented a way to demonstrate keyframes iteratively to the robot, but did not present a way to demonstrate trajectories iteratively. In this chapter, we present a method for demonstrating trajectories iteratively. The method presented here might be of interest to future usability studies such as the one presented in [53].
The idea of incremental imitation learning has also been used in conjunction with tactile feedback to allow humans to teach grasp positioning and adaptation to robots [54, 55].
As explained in Section 3.3.2, we have been using as part of our algorithm the framework of Probabilistic Movement Primitives (ProMPs) [35] for achieving generalization. This framework offers a straightforward way of inferring trajectories given contexts. Other methods such as task parameterized models discussed in [56] and [57] have very good generalization properties as well. In task parameterized models, demonstrations are observed from different frames. Different contexts are then represented by different positions and orientations of these frames. For example, by changing the position and orientation of an object of interest in the workspace of the robot, the position and orientation of the frame associated with this object also change. The product of distributions over trajectories observed from different frames determines the trajectories that should be executed in a new context. The novelty of our work resides not in the generalization to different contexts in itself but in offering the human an intuitive way to teach trajectories to a robot in an incremental manner. The human feedback also changes how the robot responds to different contexts. The incremental learning aspect of this work could be combined with task parameterized models as well.
Reinforcement Learning methods, as in [58], have been used to make robots find successful movements for solving a given task and to generalize those movements to new situations. However, in the absence of a good model of the robot and of its environment, which would allow for optimization in simulation, it might take too long to find a successful policy with the real robot. In this case, our method may be helpful by allowing the human to influence the search for good policies.

3.3 Incremental Imitation Learning

In this section, our method for incremental imitation learning of context-dependent motor skills is explained. First, in Section 3.3.1, a procedure is explained that allows a human to teach a trajectory to a robot through kinesthetic teaching and to incrementally introduce adjustments to the trajectory. Afterward, Section 3.3.2 explains how this procedure can be combined with a probabilistic framework to provide the robot with the capability of learning context-dependent motor skills from human demonstrations and incremental refinements.

3.3.1 Incremental Imitation Learning of a Trajectory for a Single Context

The workflow depicted in Fig. 3.2 describes our procedure to incrementally teach a trajectory to the robot for a single context. First, an initial desired trajectory τ_D is defined. This initial desired trajectory could have been defined by a vector of Cartesian or joint positions entered by the user or by kinesthetic teaching. We assume the existence of a feedback controller with potentially imperfect dynamics compensation. The robot tries to track τ_D, but executes in general a different trajectory τ_R. The sequence of forces or torques u_R generated by the controller at each time step is recorded. The robot returns then to its initial position and u_R is executed. While the robot executes u_R, the human can disturb the trajectory of the robot. The resulting trajectory τ_{HR} is recorded. We propose an update of the form

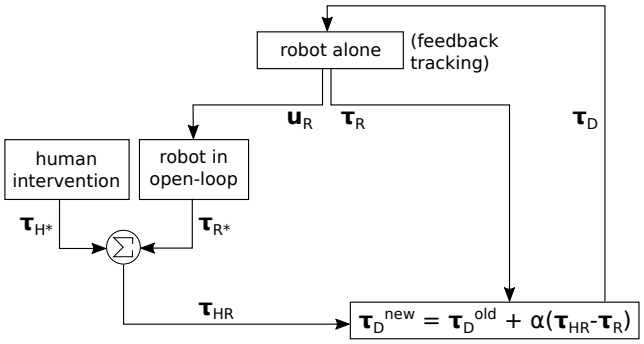


Figure 3.2.: Workflow of procedure to incrementally teach a trajectory to a robot for a single context.

$$\tau_D^{new} = \tau_D^{old} + \alpha(\tau_{HR} - \tau_R), \tag{3.1}$$

where $\alpha \in [0, 1]$ is a scalar that defines how much the difference between the trajectories τ_{HR} and τ_R should change the desired trajectory τ_D. In our experiments, the term α has been set equal to 1. Smaller values of α decrease the update step, what might be useful to adjust for cases in which the human tends to exaggerate his feedback.
The update (3.1) resembles the format of iterative learning control (ILC) [59, 60]. Fundamentally, ILC learns tracking commands with respect to a given trajectory that the robot should execute. The challenge addressed by our method

relates to the acquisition of the desired trajectory updates, which are assumed to be implicitly provided by human interventions. As in ILC, the control signal, in this case the desired trajectory τ_D for the feedback controller, is updated according to an error, in our case $(\tau_{HR} - \tau_R)$. In other words, the difference between the trajectory that the robot executes without disturbance τ_R and the trajectory that the robot executes with human disturbance τ_{HR} indicates how the control signal τ_D should be changed in order to perform the movement that the human wishes. While in ILC the reference trajectory used in the computation of the tracking error is predefined, in update (3.1) the reference trajectory τ_{HR} changes at each iteration when the human is allowed to disturb the trajectory of the robot. The human decides when to stop disturbing the trajectory of the robot, in which case $\tau_{HR} = \tau_R$ and $\tau_D^{new} = \tau_D^{old}$, which means that the desired trajectory does not change anymore.

3.3.2 Learning Context-Dependent Motor Skills from Incremental User Feedback

This section presents our method for incremental imitation learning of context-dependent motor skills. This method is based on the procedure explained in the previous section and uses Probabilistic Movement Primitives (ProMPs) [35].

3.3.2.1 Probabilistic Movement Primitives

A ProMP is a probability distribution over trajectories. This probability distribution can be built from multiple demonstrated trajectories. In the ProMP framework, each demonstrated trajectory, which has a duration of T time steps, is approximated by a weighted sum of N normalized Gaussian basis functions. This approximation can be represented by the equation

$$\boldsymbol{\tau} = \boldsymbol{\Psi}\boldsymbol{w} + \boldsymbol{\epsilon}, \tag{3.2}$$

where $\boldsymbol{\epsilon}$ is a zero-mean i.i.d. Gaussian noise, i.e. $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \boldsymbol{I}_{TxT}\right)$, and

$$\boldsymbol{\Psi} = \begin{bmatrix} \psi_1(1) & \psi_2(1) & \cdots & \psi_N(1) \\ \psi_1(2) & \psi_2(2) & \cdots & \psi_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(T) & \psi_2(T) & \cdots & \psi_N(T) \end{bmatrix}. \tag{3.3}$$

The terms $\psi_n(t)$ correspond to basis functions indexed by n and evaluated at time t. The centers of those basis functions are positioned at regular intervals along the time axis. The vector of weights $\boldsymbol{w} = [w_1, w_2, \cdots, w_N]^\top$, containing the weight w_n for each basis function ψ_n, can be computed through linear least squares, according to

$$\boldsymbol{w} = \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^\top \boldsymbol{\tau}. \tag{3.4}$$

Once the weight vector $\boldsymbol{w}$ for each demonstrated trajectory $\boldsymbol{\tau}$ has been computed, a probability distribution $p(\boldsymbol{w})$ over weight vectors can be defined. In this work, $p(\boldsymbol{w})$ is assumed to be a Gaussian with mean $\boldsymbol{\mu}_w$ and covariance $\boldsymbol{\Sigma}_w$, i.e.

$$p(\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right). \tag{3.5}$$

With this assumption, it is possible to compute in closed form the probability distribution $p(\boldsymbol{\tau})$ over trajectories by integrating out the weight vectors $\boldsymbol{w}$ as in the equation

$$p(\boldsymbol{\tau}) = \int p(\boldsymbol{\tau}|\boldsymbol{w})\, p(\boldsymbol{w})\, d\boldsymbol{w}, \tag{3.6}$$

which results in

$$p(\boldsymbol{\tau}) = \mathcal{N}\left(\boldsymbol{\mu}_\tau, \boldsymbol{\Sigma}_\tau\right), \tag{3.7}$$

where

$$\boldsymbol{\mu}_\tau = \boldsymbol{\Psi}\boldsymbol{\mu}_w, \tag{3.8}$$

$$\boldsymbol{\Sigma}_\tau = \sigma^2 \boldsymbol{I}_{TxT} + \boldsymbol{\Psi}\boldsymbol{\Sigma}_w\boldsymbol{\Psi}^\top. \tag{3.9}$$

3.3.2.2 Modeling Context-Dependent Motor Skills

In this work, ProMPs are used to infer the most probable trajectory for a given context. In order to achieve this, a normal joint probability distribution over weight vectors $\boldsymbol{w}$ and the corresponding contexts, here represented by the vectors $\boldsymbol{c}$, is created,

$$p(\boldsymbol{w},\boldsymbol{c}) = \mathcal{N}\left(\boldsymbol{\mu}_{joint}, \boldsymbol{\Sigma}_{joint}\right), \tag{3.10}$$

where

$$\boldsymbol{\mu}_{joint} = \begin{bmatrix} \boldsymbol{\mu}_{w} \\ \boldsymbol{\mu}_{c} \end{bmatrix}, \boldsymbol{\Sigma}_{joint} = \begin{bmatrix} \boldsymbol{\Sigma}_{ww} & \boldsymbol{\Sigma}_{wc} \\ \boldsymbol{\Sigma}_{cw} & \boldsymbol{\Sigma}_{cc} \end{bmatrix}.$$

Given a specific context $\boldsymbol{c}$, it is then possible to compute the conditional probability distribution

$$p(\boldsymbol{w}|\boldsymbol{c}) = \mathcal{N}\left(\boldsymbol{\mu}_{w|c}, \boldsymbol{\Sigma}_{w|c}\right), \tag{3.11}$$

where

$$\boldsymbol{\mu}_{w|c} = \boldsymbol{\mu}_{w} + \boldsymbol{\Sigma}_{wc}\boldsymbol{\Sigma}_{cc}^{-1}\left(\boldsymbol{c} - \boldsymbol{\mu}_{c}\right), \tag{3.12}$$

$$\boldsymbol{\Sigma}_{w|c} = \boldsymbol{\Sigma}_{ww} - \boldsymbol{\Sigma}_{wc}\boldsymbol{\Sigma}_{cc}^{-1}\boldsymbol{\Sigma}_{cw}. \tag{3.13}$$

Next, the conditional probability distribution $p(\boldsymbol{\tau}|\boldsymbol{c})$ can be computed by solving the equation

$$p(\boldsymbol{\tau}|\boldsymbol{c}) = \int p(\boldsymbol{\tau}|\boldsymbol{w})\, p(\boldsymbol{w}|\boldsymbol{c})\, d\boldsymbol{w}, \tag{3.14}$$

which results in

$$p(\boldsymbol{\tau}|\boldsymbol{c}) = \mathcal{N}\left(\boldsymbol{\mu}_{\tau|c}, \boldsymbol{\Sigma}_{\tau|c}\right), \tag{3.15}$$

with

$$\boldsymbol{\mu}_{\tau|c} = \boldsymbol{\Psi}\boldsymbol{\mu}_{w|c}, \tag{3.16}$$

$$\boldsymbol{\Sigma}_{\tau|c} = \sigma^2 \boldsymbol{I}_{TxT} + \boldsymbol{\Psi}\boldsymbol{\Sigma}_{w|c}\boldsymbol{\Psi}^{\top}. \tag{3.17}$$

3.3.2.3 Online Learning with Human Feedback

This section explains the proposed algorithm, which allows robots to learn in an online fashion context-dependent motor skills from human demonstrations and refinement. This algorithm uses the refinement procedure illustrated as a workflow in Fig. 3.2 in conjunction with the probabilistic modeling of context-dependent motor skills previously explained in Section 3.3.2.2.

The robot starts with an initial joint probability distribution $p(\boldsymbol{w},\boldsymbol{c})$ over weights $\boldsymbol{w}$ and contexts $\boldsymbol{c}$. Given this prior and a certain context $\boldsymbol{c}$, the robot computes $p(\boldsymbol{\tau}|\boldsymbol{c}) = \mathcal{N}\left(\boldsymbol{\mu}_{\tau|c}, \boldsymbol{\Sigma}_{\tau|c}\right)$, as in Section 3.3.2.2. The robot's desired trajectory $\boldsymbol{\tau}_D$ is set equal to the mean $\boldsymbol{\mu}_{\tau|c}$. The algorithm iterates over the refinement loop depicted in Fig. 3.2 as many times as the human wants. By the end of this iteration, the weights $\boldsymbol{w}$ of the new desired trajectory $\boldsymbol{\tau}_D$ are computed, using (3.4). This new weight vector $\boldsymbol{w}_M$ and the given context vector $\boldsymbol{c}_M$ are concatenated to form the vector $\boldsymbol{x} = \left[\boldsymbol{w}_M^{\top}, \boldsymbol{c}_M^{\top}\right]$, where M is the number of situations experienced so far. The joint distribution $p(\boldsymbol{w},\boldsymbol{c}) = \mathcal{N}\left(\boldsymbol{\mu}_{joint}, \boldsymbol{\Sigma}_{joint}\right)$ is then updated according to Welford's method for computing mean and covariance online [61]. According to this method, the mean $\boldsymbol{\mu}_{joint}$ is updated with

$$\boldsymbol{\mu}_{joint}^{new} = \boldsymbol{\mu}_{joint}^{old} + \frac{\left(\boldsymbol{x}^{\top} - \boldsymbol{\mu}_{joint}^{old}\right)}{M} \tag{3.18}$$

and the covariance matrix $\boldsymbol{\Sigma}_{joint}$ with

$$\begin{aligned} &\boldsymbol{S}^{new}(i,j) = \boldsymbol{S}^{old}(i,j) + \\ &\left(\frac{M-1}{M}\right)\left(\boldsymbol{x}(i) - \boldsymbol{\mu}_{joint}^{old}(i)\right)\left(\boldsymbol{x}(j) - \boldsymbol{\mu}_{joint}^{old}(j)\right), \end{aligned} \tag{3.19}$$

$$\boldsymbol{\Sigma}_{joint}^{new}(i,j) = \frac{\boldsymbol{S}^{new}(i,j)}{M-1}, \tag{3.20}$$

where $\boldsymbol{S} = (M-1)\boldsymbol{\Sigma}$ is an auxiliary matrix.

Afterward, the whole procedure is repeated for a new context. Algorithm 1 summarizes the proposed method.

Algorithm 1 Incremental Imitation Learning of Context-Dependent Motor Skills

Initialize $\boldsymbol{\mu}_{joint}$ and $\boldsymbol{\Sigma}_{joint}$ with a few demonstrations or with predefined values
for each new $\boldsymbol{c}$
    compute $\boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{c}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{c}}$ (Eqs. 3.12 and 3.13)
    compute $\boldsymbol{\mu}_{\tau|\boldsymbol{c}}$ and $\boldsymbol{\Sigma}_{\tau|\boldsymbol{c}}$ (Eqs. 3.16 and 3.17)
    $\boldsymbol{\tau}_D = \boldsymbol{\mu}_{\tau|\boldsymbol{c}}$
    refinement loop (until human decides to stop)
        robot tracks $\boldsymbol{\tau}_D$ with feedback controller ($\boldsymbol{\tau}_R$ and $\boldsymbol{u}_R$ are recorded)
        robot executes $\boldsymbol{u}_R$ in feedforward and human is allowed to disturb the trajectory ($\boldsymbol{\tau}_{HR}$ is recorded)
        $\boldsymbol{\tau}_D^{new} = \boldsymbol{\tau}_D^{old} + \alpha(\boldsymbol{\tau}_{HR} - \boldsymbol{\tau}_R)$
    end
    $\boldsymbol{w}_M = \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^\top \boldsymbol{\tau}_D$
    update $\boldsymbol{\mu}_{joint}$ and $\boldsymbol{\Sigma}_{joint}$ (Eqs. 3.18, 3.19 and 3.20)
end

3.4 Experiments

This section presents two experimental evaluations of the proposed algorithm. First, a simple 2D problem is presented. In this problem, the human can incrementally teach trajectories to the learning system, which can infer what to do in the face of new contexts. Afterward, experiments are described that show the applicability of our method for teaching motor skills to a real robot.
In all the experiments described in this chapter, the number N of Gaussian basis functions, introduced in Section 3.3.2.1, was 20. This value was determined empirically by increasing the number of basis functions until the trajectories could be approximated to a desired level of detail.

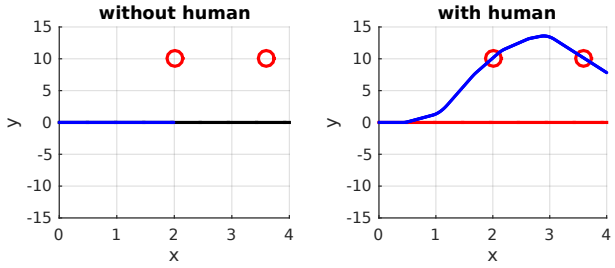


Figure 3.3.: 2D problem. **Left plot**: the red circles represent two via points through which the trajectory should pass; the black line represents the trajectory being currently tracked by the particle; the blue line represents the trajectory executed by the particle until the current instant. **Right plot**: the red circles represent the same two desired via points; the red line represents the trajectory that the particle would execute without any human interference; the blue curve represents the trajectory executed by the particle with human interference.

3.4.1 2D Problem

We designed a simple 2D problem in order to facilitate the understanding of the algorithm and the visualization of results. In this problem, a particle moves with constant velocity in the x-direction. Initially, its velocity in the y-direction is zero. The user can introduce an acceleration in the y-direction by pressing the keys “up” or “down” on the keyboard. The objective is to teach the system a trajectory that passes through two via points. The y-coordinates of each via point constitute in this case the context, which is represented by the vector $[y_1, y_2]^\top$, where y_1 is the y-coordinate of the via point located at $x = 2$ and y_2 is the y-coordinate of the via point located at $x = 3.6$.
After being trained for a number of different configurations of the via points, the system should be able to infer the right trajectory to pass through the via points both in the configurations that it has already experienced as well as in a previously unseen configuration. The human can iteratively apply disturbances to the trajectory of the particle. This way, the human can correct the trajectory initially inferred by the system for a given context. See Fig. 3.3 for an illustration of this 2D problem.
In this problem, $\boldsymbol{u}_R$ is a time-indexed sequence of forces in y direction, while the trajectories $\boldsymbol{\tau}_R$, $\boldsymbol{\tau}_{HR}$ and $\boldsymbol{\tau}_D$ are time-indexed sequences of (x, y) positions.
Fig. 3.4 shows the prior and the posterior probability distributions over trajectories after the human had trained the system for two contexts. In this simple problem, after experiencing only two examples, the system could already infer trajectories that passed close to the via points. The progress in the generalization ability of the system is depicted in Figs. 3.5 and 3.6. Fig. 3.5 shows the prior and the posterior over trajectories without any training, after the human had trained the system for one context and so on until the system had previously observed eight different contexts. Fig. 3.6 shows the mean squared error

$$MSE = \frac{1}{2}\left((y(2) - y_1)^2 + (y(3.6) - y_2)^2\right) \tag{3.21}$$

of the trajectories executed by the particle when observing each context for the first time. Here $y\,(2)$ and $y\,(3.6)$ are the y-coordinates of the particle when $x = 2$ and $x = 3.6$ respectively.
The MSE for the first context is 100 because the trajectory is 10 units apart from each via point. After the human has taught the trajectory for two different contexts through the refinement loop, the MSE of the trajectories executed by the particle for previously unseen contexts already drops to almost zero.

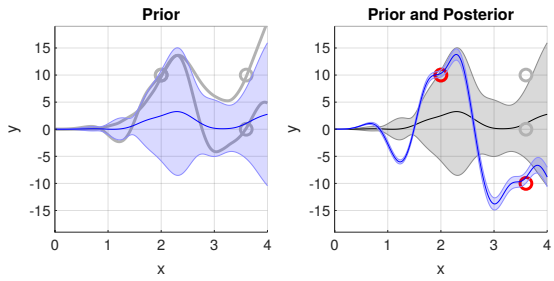


Figure 3.4.: Prior and posterior over trajectories for the 2D problem after the human had trained the system for two contexts. **Left plot**: the light gray curves represent the trajectories executed by the particle after refinement provided by the human for two different configurations of the via points, represented by the light gray circles; the blue curve represents the prior mean and the blue shade represents two standard deviations of the prior. **Right plot**: the configuration of the via points represented by the red circles is different from the configurations that have been observed so far by the system, which are represented by the light gray circles; the black curve represents the prior mean and the gray shade represents two standard deviations of the prior (the prior in this plot is the same as in the left plot); the blue curve represents the posterior mean and the blue shade represents two standard deviations of the posterior.

3.4.2 Real Robot Experiments

We tested our algorithm in experiments with the BioRob [48], a 4-DoF elastically actuated robot arm. It is a biologically inspired robot whose cables and springs roughly mimic the functionality of antagonistic pairs of muscles. The BioRob is light and compliant, nevertheless challenging to model and control[1].
In these experiments, the objective of the human was to teach the robot how to reach a ball positioned on steps of a ladder as shown in Fig. 3.1. We defined six different positions for the ball (see Fig. 3.7). We chose to represent each position by two values, expressing the two different heights with respect to the ground and the three different left-to-right positions. This choice was motivated by the fact that the relations between the values assumed by the context variables are important, not their absolute values. In this setup, there was no camera detecting the ball. The values representing the different contexts were manually provided to the robot.
In our experiments with the BioRob, $\boldsymbol{u}_R$ is a time-indexed sequence of torques, while the trajectories $\boldsymbol{\tau}_R$, $\boldsymbol{\tau}_{HR}$ and $\boldsymbol{\tau}_D$ are time-indexed sequences of joint angles. By working with trajectories in joint space, we avoid any possible problems related to kinematic redundancy that would need to be addressed if our trajectories were sequences of Cartesian end-effector positions.
In order to initialize the prior probability distribution $p(\boldsymbol{w}, \boldsymbol{c})$ over weight vectors and contexts, the refinement loop depicted in Fig. 3.2 was executed for four different positions of the ball: "top left" represented by the vector $[1,1]^\top$, "top right" represented by $[1,3]^\top$, "down left" represented by $[2,1]^\top$ and "down right" represented by $[2,3]^\top$.
After building the prior, the for loop described in Algorithm 1 was executed. The robot computed the most probable trajectory $\boldsymbol{\tau}$ for the context "top middle" represented by $[1,2]^\top$. The human improved the trajectory of the robot through the refinement loop until the robot could successfully reach the ball at the "top middle" position. After the human decided to quit the refinement loop, the joint probability distribution $p(\boldsymbol{w}, \boldsymbol{c})$ over weights and contexts was updated according to (3.18), (3.19) and (3.20). Then the robot used its updated prior to compute a trajectory for the context "down middle" represented by $[2,2]^\top$.
Having built the prior based on trajectories that reach the ball at each of the four corner positions, the robot was able to compute trajectories to reach or pass close to the ball at the two previously untrained middle positions. Using our refinement loop, the human could correct the trajectories inferred by the robot. In a second pass of the algorithm over all the six positions, the robot was able to reach the ball every time without needing any further refinement from the human.
Fig. 3.8 shows three iterations of the refinement loop to teach the robot how to reach the ball at the position "top left" when the robot had no previous experience. The figure shows the trajectories in joint space of the 1^{st} DoF of the BioRob and the corresponding end-effector trajectories in Cartesian space. In the first iteration, the robot arm simply hung down. Then, the human demonstrated through kinesthetic teaching how to reach the ball. In the second iteration, the robot tried to track the new desired trajectory. Since the controller of the robot is not able to track a desired trajectory accurately, the robot only passed close to the ball, but still did not reach it (see the middle plot in the lower row). The

[1] The robot is controlled by a proportional-derivative (PD) controller with feedforward terms to compensate for gravity and friction/stiction. In these experiments, only three DoFs are actually being used.

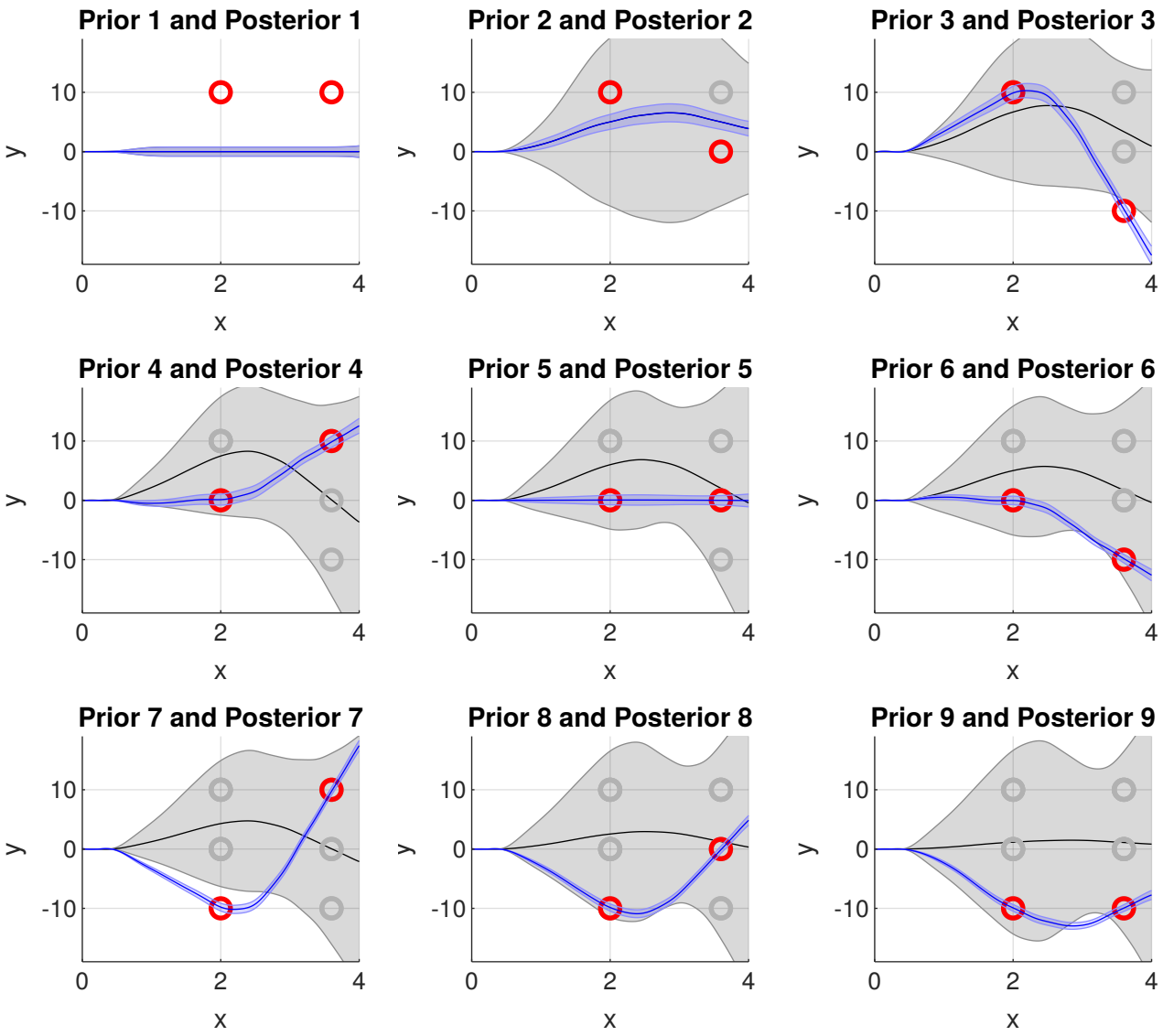


Figure 3.5.: Prior and posterior (with mean and two standard deviations) over trajectories given contexts without any training (Prior 1 and Posterior 1), after training for one context (Prior 2 and Posterior 2) and so on until the system had previously observed eight different contexts (Prior 9 and Posterior 9). Only the two first contexts needed refinement from the human. After that, the system was already able to infer trajectories intercepting the via points.

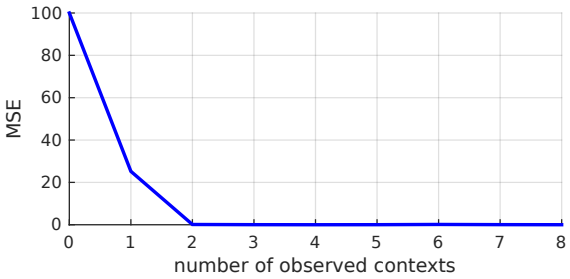


Figure 3.6.: Mean squared error (MSE) of trajectory executed by the particle when the system is observing each context for the first time.

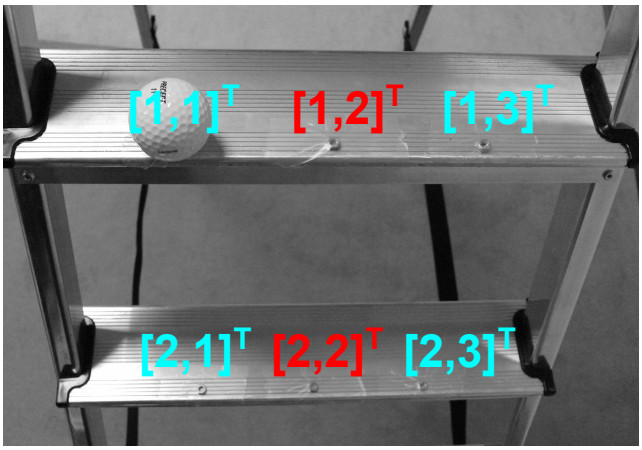


Figure 3.7.: Ball positions in an experiment involving the real robot. The positions represented by the vectors written in blue are the contexts that were used to initialize the prior probability distribution $p(\boldsymbol{w}, \boldsymbol{c})$ over weight vectors and contexts. Having built this prior, the robot infers trajectories to reach the ball at the positions represented by the vectors written in red.

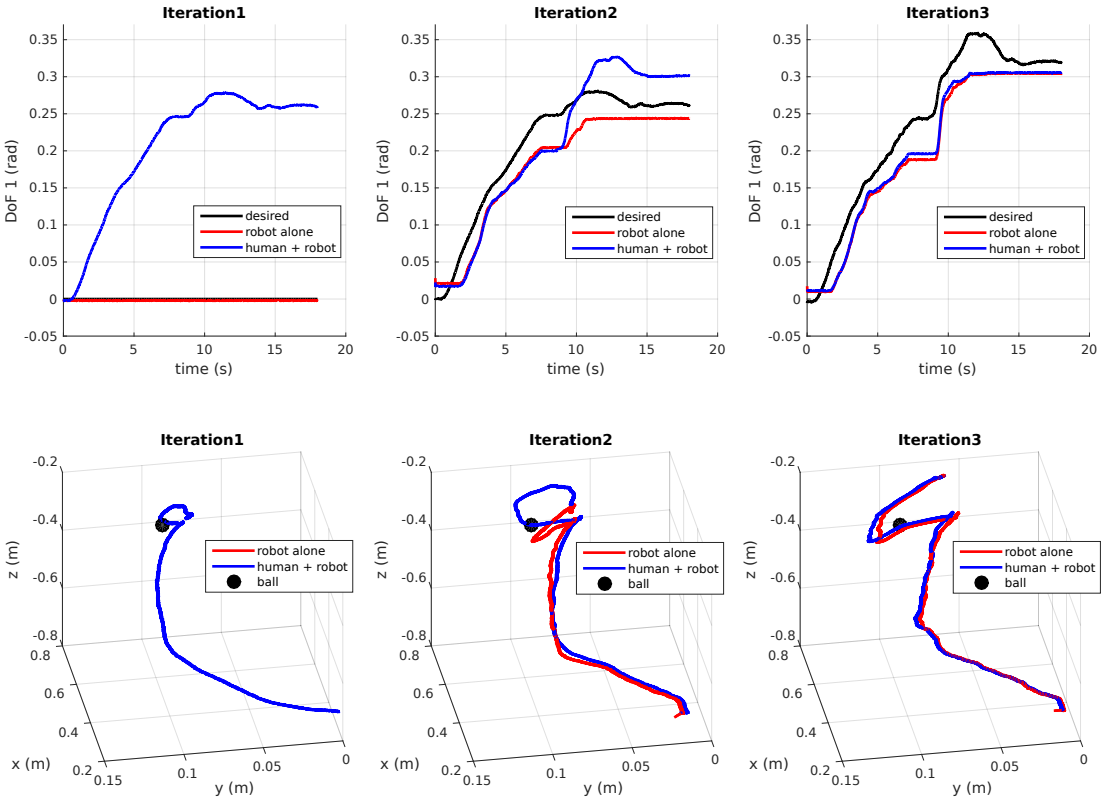


Figure 3.8.: Three iterations of the refinement loop. **Upper row**: Trajectories in joint space of the 1^{st} DoF of the BioRob. **Lower row**: The corresponding end-effector trajectories in Cartesian space.

human refined this trajectory further by interfering in the movement of the robot. In the third iteration, the robot was able to reach the ball alone and the human did not give any further feedback. The small differences between the red curve (trajectory generated by feedback tracking) and the blue curve (trajectory generated by reproducing the sequence of torques generated by the feedback tracking controller) are, in the third iteration, due to small errors in the repeatability of the robot.
We also performed experiments in which the human taught a robot how to putt in a minigolf-like task. In this case, there was only one context. In the phases in which the human was allowed to disturb the trajectory, the human kept his hands close to the robot and applied forces to its end-effector when he judged necessary. After a few iterations, the desired trajectory τ_D tracked by the feedback controller was such that it led to the robot being able to sink the ball.

3.5 Epilogue

This chapter presented an algorithm to allow humans to incrementally teach robots context-dependent motor skills. This algorithm is particularly relevant when the robot cannot track desired trajectories accurately or when trajectories initially computed by the robot given a new context do not solve the task at hand. In those cases, our algorithm offers the human an intuitive way of refining the trajectories of the robot. Moreover, the refined trajectories and the new contexts are used to update the probability distribution used by the robot to compute trajectories given contexts. A 2D problem and experiments involving a 4-DoF elastically actuated robot arm demonstrate the effectiveness of the proposed algorithm.
As it is, the presented method for the human to teach the robot is not suitable when the robot moves very fast, is too heavy or manipulates a dangerous object. For those cases, an alternative way for the human to introduce changes in the trajectory of the robot would be necessary. Instead of physically interacting with the robot while it moves, the human could for instance use a graphical interface to change the trajectory or use teleoperation.
In this work, we modeled the joint probability distribution $p(\boldsymbol{w}, \boldsymbol{c})$ over weights and context variables as a single Gaussian. This model entails that $\boldsymbol{w}$ and $\boldsymbol{c}$ are linearly correlated, which is a reasonable assumption for the simple tasks we have dealt with so far. On the other hand, in a task such as playing table tennis, in which the robot would have to execute forehand and backhand strokes, this assumption would probably not hold. In order to deal with those cases, an alternative would be to model $p(\boldsymbol{w}, \boldsymbol{c})$ as a Gaussian mixture model.
The contexts addressed so far in our work have been only in the form of via points. Other possible contexts could be the weight or the size of objects manipulated by the robot, goal position of an object thrown by the robot, positions and orientations of objects in the workspace, etc. For the simple contexts evaluated so far, the human could teach the robot in a few iterations how to solve the task at hand and the generalization capabilities of the algorithm also helped to reduce the amount of human intervention needed. Further evaluations shall be made in order to determine if the human can successfully teach skills with other types of context as well.

4 Movement Primitives with Multiple Phase Parameters

Movement primitives are concise movement representations that can be learned from human demonstrations, support generalization to novel situations and modulate the speed of execution of movements. The speed modulation mechanisms proposed so far are limited though, allowing only for uniform speed modulation or coupling changes in speed to local measurements of forces, torques or other quantities. Those approaches are not enough when dealing with general velocity constraints. We present a movement primitive formulation that can be used to non-uniformly adapt the speed of execution of a movement in order to satisfy a given constraint while maintaining similarity in shape to the original trajectory. We present results using a 4-DoF robot arm in a minigolf setup.

4.1 Prologue

Learning from human demonstrations has been playing an increasingly important role in robotics, especially since robots are getting better at adapting demonstrated movements to new situations and demands [62–64].

A commonly required adaptation is due to the necessity of executing a certain demonstrated movement faster or slower. For example, a robot playing table tennis has to hit the ball at different speeds than the ones of the movements demonstrated to the robot [63]. In a task such as golf, the necessity of adapting a certain movement in order to achieve new speeds also arises. Depending on the distance between the ball and the hole or on the shape and friction properties of the floor, a robot has to hit the ball faster or slower in order to sink it.

When executing a golf swing, a human normally moves the golf club first away from the ball and then towards it. In order to hit the ball twice as fast, the human does not need to move away from the ball twice as fast as well. Uniformly accelerating the whole movement would be energy inefficient. Therefore, adapting a golf swing to hit the ball with different speeds requires a non-uniform change in the speed of execution of the movement.

This chapter presents an approach to achieve such non-uniform changes in speed of execution in order to adapt a demonstrated movement to new situations and demands. This approach consists of optimizing with respect to a given reward multiple parameters that define the phase function associated with the movement. The phase function is a monotonically increasing function of time. It describes the speed of execution of the movement at each time step and its overall duration.

Figure 4.1.: Human demonstrating a putt to a robot via kinesthetic teaching.

The remainder of this chapter is organized as follows: Section 4.2 presents related work. Section 4.3 explains our movement primitive formulation with multiple phase parameters. Section 4.4 shows how to use a reinforcement learning algorithm to optimize the parameters of the phase function and the amplitude of a movement. Section 4.5 presents two sets of experiments. The first experiments explain the importance of non-uniformly adapting the phase function of a movement instead of simply adjusting some parameters that define its shape or uniformly rescaling its speed of execution. The second set of experiments has been conducted with an elastically actuated robot arm and shows how the proposed approach performs in practice at the task of adapting a demonstrated golf swing to achieve desired speeds at specific positions. Section 4.6 presents conclusions and ideas for future work.

4.2 Related Work

A number of distinct movement representations have been proposed that allow for adapting demonstrations to new requirements. Dynamical movement primitives (DMPs) [65], for instance, allow for changing the goal, the speed and the

duration of a movement while preserving its overall shape. The original DMP formulation, however, is not able to reach a certain goal with an arbitrary velocity because it enforces zero velocity at the end of the movement.
Kober et al. [66] have designed an alternative DMP formulation to achieve arbitrary velocities at the end of the movement, however, as pointed out by Mülling et al. [63], this solution is not necessarily accurate and may produce very large accelerations if the desired final position and the start position are very close to each other.
Mülling et al. [63] have used a two-stage movement primitive in order to achieve the desired velocity at an intermediary position along the movement. In a table tennis setup, the stage of the movement primitive is switched when the racket gets in contact with the ball. In our work, it is possible to specify desired velocities at intermediary positions using one single movement primitive with one single stage.
Nemec et al. [67] also proposed a method for adapting the velocity of motor skills learned from user demonstration. They have extended DMPs with a scaling factor $v(x)$, which is a function of the phase variable x. By changing $v(x)$, it is possible to accelerate or decelerate movements in order to satisfy for example certain contact forces or contact torques during the execution of a task. This acceleration or deceleration does not need to be uniform. In their method, the change in $v(x)$ depends on the difference between desired and actual forces and torques at successive time steps along the execution of the trajectory.
In [43], a similar approach has been adopted in order to speed up the movement of a robot carrying a cup without spilling the water inside it. In this approach, DMPs have been extended with a scaling factor $v(t)$, which allows for non-uniform change in speed. The change in $v(t)$ depends on an intermediate cost function with value γ if the water is about to be spilled or value 0 otherwise.
In contrast, our method deals with the problem of accelerating, decelerating or changing the amplitude of trajectories in order to satisfy velocity constraints. Our method allows for non-uniform change in velocity also if there is only a final reward, computed at the end of the execution of a trajectory.
In [45], van den Berg et al. showed how to speed up the movement of a surgical robot by increasing a factor s that rescales the time steps according to $\Delta t = \Delta t_0 / s$. By increasing s, the time steps get shorter. Since the number of time steps N is kept constant, the speed of the movement increases uniformly and its duration decreases. In our work, we deal with changing the speed of execution of a movement in a non-uniform fashion. As previously discussed, when adapting movements such as a golf putt swing to achieve new desired velocities at specific positions, a non-uniform change in the speed of execution is more suitable.
Englert et al. [42] presented a method for adapting online the path and the phase of a movement to events such as changes in the positions of obstacles and of the goal. Kim et al. [68] have also developed a method for adapting online the velocity of the movement of a robot to make it reach a certain position at a certain time, for example when catching objects on the fly. In contrast, we try to optimize the phase function and the amplitude of a movement according to a given reward, which can depend for instance on the difference between the achieved velocity and the desired velocity at a specific position.
In a remarkable work involving a minigolf setup, Kronander et al. [69] proposed a method to infer the hitting speed and hitting angle to sink the ball given multiple successful demonstrations. In our experiments, the amplitude of the movement and the phase function with multiple parameters have been optimized starting from only one demonstration.

4.3 Movement Primitive with Multiple Parameters for Shape and Phase

This section explains the proposed movement primitive formulation. This formulation comprises a set of parameters to define the shape of the movement in space and a set of parameters to define its speed profile and duration.

4.3.1 Shape Parameterization

The shape parameterization used in this work is basically the same one used in Probabilistic Movement Primitives (ProMPs) [35]. A trajectory $\boldsymbol{\tau} = [q_1, q_2, \cdots, q_T]^\top$ comprising positions q_t sampled at a certain number T of time steps can be approximated by a weighted sum of normalized Gaussian basis functions ψ_n. This approximation can be expressed by

$$\boldsymbol{\tau} \approx \boldsymbol{\Psi} \boldsymbol{w}, \tag{4.1}$$

where $\boldsymbol{w} = [w_1, w_2, \cdots, w_N]^\top$ is the vector of weights w_n for each normalized Gaussian basis function ψ_n and

$$\boldsymbol{\Psi} = \begin{bmatrix} \psi_1(z(1)) & \psi_2(z(1)) & \cdots & \psi_N(z(1)) \\ \psi_1(z(2)) & \psi_2(z(2)) & \cdots & \psi_N(z(2)) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(z(T)) & \psi_2(z(T)) & \cdots & \psi_N(z(T)) \end{bmatrix} \tag{4.2}$$

is a matrix with each ψ_n evaluated over the phase $z(t)$, which is a monotonically increasing function of time.
Given a trajectory $\boldsymbol{\tau}$ and a matrix of normalized Gaussian basis functions $\boldsymbol{\Psi}$, the vector of shape parameters $\boldsymbol{w}$ can be determined by

$$\boldsymbol{w} = \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^\top \boldsymbol{\tau}, \tag{4.3}$$

which is the ordinary least squares solution for linear regression [47].

4.3.2 Phase Parameterization

The phase function $z(t)$ is a monotonically increasing function of time assuming non-negative real values between 0 and Z. In this work, $Z = 100$ has been chosen. When $z(t) = 0$, the movement is just starting. When $z(t) = Z$, the movement is over. We propose defining the phase function with a few parameters that will be useful to manipulate the evolution of a movement in time, as the shape parameters w_n allow for manipulating the trajectory of a movement in space.
The rate of change of phase in relation to time can be written as the vector $\dot{\boldsymbol{z}} = [\dot{z}(1), \dot{z}(2), \cdots, \dot{z}(T-1)]^\top$. In fact, we parameterize $\dot{\boldsymbol{z}}$ according to

$$\dot{\boldsymbol{z}} = \boldsymbol{\Phi}\boldsymbol{\alpha}. \tag{4.4}$$

The matrix $\boldsymbol{\Phi}$ comprises M normalized Gaussian basis functions evaluated from time step $t = 1$ until time step $t = T-1$, where T is the total number of time steps of the movement. This matrix can be written as

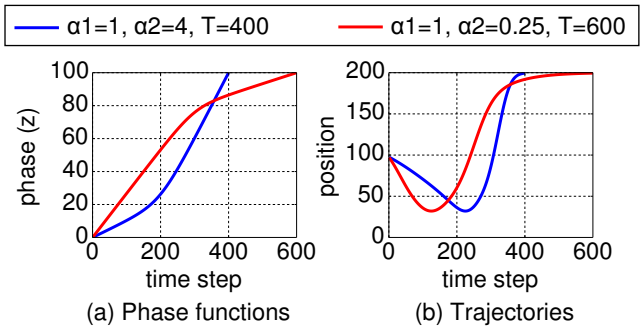


Figure 4.2.: Effects of changes in phase parameters. (a) Two different phase functions. The blue one has phase parameters $\alpha_1 = 1$, $\alpha_2 = 4$ and $T = 400$. The red one has phase parameters $\alpha_1 = 1$, $\alpha_2 = 0.25$ and $T = 600$. (b) Respective trajectories.

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_1(1) & \phi_2(1) & \cdots & \phi_M(1) \\ \phi_1(2) & \phi_2(2) & \cdots & \phi_M(2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(T-1) & \phi_2(T-1) & \cdots & \phi_M(T-1) \end{bmatrix}. \tag{4.5}$$

The vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_M]^\top$ contains the weights α_m for each basis function ϕ_m. Those weights are such that $\alpha_m > 0, \forall m \in \{1, 2, \cdots, M\}$, ensuring that the phase z always increases with time.
Once $\dot{\boldsymbol{z}}$ has been defined according to (4.4), the phase $z(t)$ can be computed via Euler integration with $z(0) = 0$ and $z(t+1) = z(t) + \Delta t \dot{z}(t)$ until $t = T-1$.
In order to ensure that the phase achieves its maximum value exactly at the last time step of the movement, i.e. $z(T) = Z$, the phase is normalized with

$$\boldsymbol{z}_{\text{norm}} = \frac{\boldsymbol{z}}{z(T)} Z. \tag{4.6}$$

In summary, the weights α_m and the total number of time steps T are the phase parameters. They define the rate of change $\dot{z}$ of the phase in relation to time, which, via Euler integration, defines the phase z of a movement.
Fig. 4.2 shows two phase functions. Each of them was defined using two normalized Gaussian basis functions $\phi(t)$. One Gaussian is centered at $t = 0$, the other at $t = T$. The variance of those Gaussians is $30 \times T$. The number of time steps T is 400 for the phase depicted in blue and 600 for the phase depicted in red. Observe how the proportion between the parameters α_m influence the phase function and consequently the evolution in time of the trajectories generated with the same shape parameters $\boldsymbol{w}$. The first half of the blue trajectory is slow compared to the second half because $\alpha_1 < \alpha_2$. The opposite happens in the red trajectory.
In [70], we proposed a similar phase function parameterization as in this chapter. In that formulation, it was necessary to define the maximum number of time steps of the demonstrated movements. By treating T as a phase parameter and applying the normalization described by (4.6) we have eliminated in this new formulation the necessity of defining a maximum number of time steps.

4.4 Reinforcement Learning of Movement Amplitude and Phase

This section explains how to use reinforcement learning by Reward-weighted Regression (RWR) [71] in order to optimize the amplitude and the phase parameters of a movement with the objective of achieving a desired velocity at a specific position.
The reward function has been defined as

$$R = \exp(-\beta \| v - v^* \|), \tag{4.7}$$

where β is a task-specific parameter tuned by the user and $\| v - v^* \|$ is the L^2 norm of the difference between v and v^*. The term v is the actual velocity at the specific position $\boldsymbol{q}^*$, while the term v^* is the desired velocity at this same specific position. Velocities have been computed using the finite difference approximation

$$\dot{\boldsymbol{q}}(t) = \frac{\boldsymbol{q}(t+1) - \boldsymbol{q}(t)}{\Delta t}. \tag{4.8}$$

The term β has been set equal to 10 in all our experiments because this value led to achieving the desired speed with high precision within a reasonable number of reinforcement learning iterations, compared to the values 0.1, 1 and 100.
The parameters that need to be optimized are given by the vector $\boldsymbol{\theta} = [\alpha_2, \alpha_3, \cdots, \alpha_M, T, \lambda]^\top$. Due to the normalization (4.6), not the absolute values of the parameters α_m, but the proportions between their values is important. Therefore, α_1 is simply always equal to 1 and it is not necessary to optimize it. The scalar λ multiplies all the shape parameters w_n and determines the amplitude of the movement.
Let us adopt an upper-level policy given by a Gaussian distribution $\mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\right)$. Our objective is thus to maximize the expected reward with respect to $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$. This can be done iteratively according to

$$\{\boldsymbol{\mu}_{\boldsymbol{\theta}}^{k+1}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{k+1}\} = \underset{\{\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\}}{\arg\max} \sum_{i=1}^{S} R_i \mathcal{N}\left(\boldsymbol{\theta}_i; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\right), \tag{4.9}$$

where S is the number of sampled parameter vectors $\boldsymbol{\theta}_i$ from the previous policy $\mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}^k, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^k\right)$.
The solution of (4.9) is given by

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}^{k+1} = \frac{\sum_{i=1}^{S} R_i \boldsymbol{\theta}_i}{\sum_{i=1}^{S} R_i}, \tag{4.10}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{k+1} = \frac{\sum_{i=1}^{S} R_i \left(\boldsymbol{\theta}_i - \boldsymbol{\mu}_{\boldsymbol{\theta}}^k\right)\left(\boldsymbol{\theta}_i - \boldsymbol{\mu}_{\boldsymbol{\theta}}^k\right)^\top}{\sum_{i=1}^{S} R_i}. \tag{4.11}$$

This iterative process continues until the expected reward converges. The best parameters are then given by $\boldsymbol{\mu}_{\boldsymbol{\theta}}$.
The convergence properties of the Expectation-Maximization (EM) algorithm [72] guarantee that the reward converges to some local maximum. The initial parameters $\boldsymbol{\mu}_{\boldsymbol{\theta}}^0$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^0$ influence the maximum expected reward achieved by RWR. As will be explained in Section 4.5, those parameters have been initialized in our work by using a human demonstration and choosing an exploration noise.

4.5 Experiments

This section presents a number of experiments in which the proposed movement primitive formulation with multiple phase parameters has been applied. In these experiments, the BioRob, a robot arm consisting of four elastically actuated joints [48], has been used.
In the first experiments, we have assumed that the desired trajectory could be perfectly tracked by the robot. Using this simplifying assumption, a comparison between optimizing in simulation different parameters has been made. Afterward, both the phase parameters and the amplitude of a putt swing have been optimized such that the real robot arm could pass through a specific position with a desired velocity.

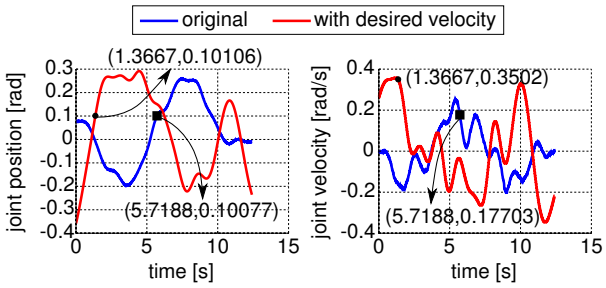


Figure 4.3.: Optimizing only shape parameters to achieve double the velocity of an original movement when the 1^{st} joint angle crosses the value $0.1\,\text{rad}$ for the first time. Assuming the desired trajectory could be exactly tracked by the robot.

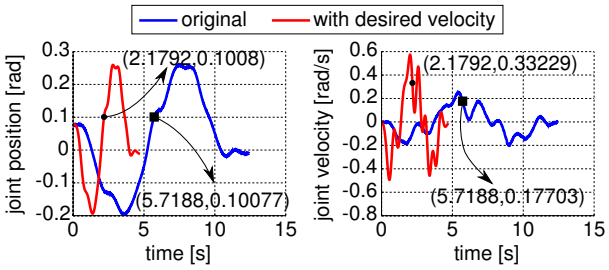


Figure 4.4.: Optimizing a phase function of the form $z(t) = \alpha t$ to achieve double the velocity of the original movement when the 1^{st} joint angle crosses the value $0.1\,\text{rad}$ for the first time. Assuming the desired trajectory could be exactly tracked by the robot.

4.5.1 Comparison between Optimizing Different Parameters

In these experiments, a putt swing was demonstrated to the robot via kinesthetic teaching as depicted in Fig. 4.1. Subsequently, the robot tried to track the demonstrated trajectory[1]. The trajectory executed by the robot was then recorded. It comprises 5959 positions[2] sampled at regular intervals of $1/480s$. We refer to this recorded trajectory as the "original trajectory".

The shape of the original trajectory has been parameterized according to (4.1). Twenty normalized Gaussian basis functions $\psi(z)$ have been used. These basis functions have variance equal to 100 and their means are evenly distributed between $z = 0$ and $z = 100$. In a first experiment, the phase function was defined as $z(t) = \alpha t$, where $\alpha = 100/T$. The vectors of shape parameters $\boldsymbol{w}$ for each of the four DoFs of the robot have been determined with (4.3).

The shape parameters have then been optimized according to (4.9), using however $\boldsymbol{\theta} = \boldsymbol{w}$, to generate a trajectory in which the double of the original velocity of the 1^{st} joint is achieved when this joint crosses position $0.1\,\text{rad}$ for the first time. The desired velocity for the other three joints is the same as the original velocity and is close to zero. Our algorithm iterated over equations (4.10) and (4.11) 100 times. In each iteration, fifty vectors of parameters $\boldsymbol{\theta}$ were sampled. Results are depicted in Fig. 4.3. The plot on the left shows the time at which the original trajectory and the trajectory after optimization cross the joint position $0.1\,\text{rad}$ for the first time. The plot on the right shows the joint velocity of both trajectories. The shape of the trajectory has changed considerably. Such changes in shape may result in unnecessary movements, collisions, reaching joint limits, etc. Further constraints would be required in order to keep the shape of the movement similar to the original one when optimizing only the shape parameter vectors $\boldsymbol{w}$ as in this example.

By optimizing not the shape parameters, but only the phase parameter α, assuming $z(t) = \alpha t$ and $\alpha = 100/T$, we have obtained solutions like the one depicted in Fig. 4.4. This solution rescales the velocity of the movement uniformly and results in unnecessary accelerations. Speeds are considerably higher than the original ones also when the joint position is far away from the position of interest.

Finally, the same experiment was performed optimizing multiple phase parameters and the amplitude of the movement, as in Section 4.4. The shape parameterization is the same as the one in the previous experiment. The phase function has been defined as in Section 4.3.2, with nine Gaussian basis functions $\phi(t)$. The basis functions $\phi(t)$ have variance equal to the duration T and their means are equally distributed between $t = 0$ and $t = T$. The upper-level policy $\mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\right)$ was initialized with mean

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}^{0} = \left[\alpha_2 = 1, \alpha_3 = 1, \cdots, \alpha_9 = 1, T = 500, \lambda = 1\right]^{\top}$$

and a diagonal covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{0}$ with its diagonal defined by the vector

$$\left[\sigma_{\alpha_2}^2 = 10, \cdots, \sigma_{\alpha_9}^2 = 10, \sigma_T^2 = 1000, \sigma_\lambda^2 = 0.1\right].$$

The elements $\sigma_{\alpha_m}^2$ represent the variance of the parameters α_m. The element σ_T^2 represents the variance of the duration T of the movement. Finally, σ_λ^2 represents the variance of the amplitude parameter λ.

[1] The control of the robot has been performed using a PD controller for joint and motor angles with compensation for stiction and gravity. System identification methods have been used to estimate gravitational torques, elastic transmission and stiction.

[2] Cubic spline interpolation has been used to compress the original trajectory to 500 time steps before determining the parameters of the movement primitive and running reinforcement learning. The optimized trajectory is then decompressed also with cubic spline interpolation in the experiments where the real robot executes the trajectories.

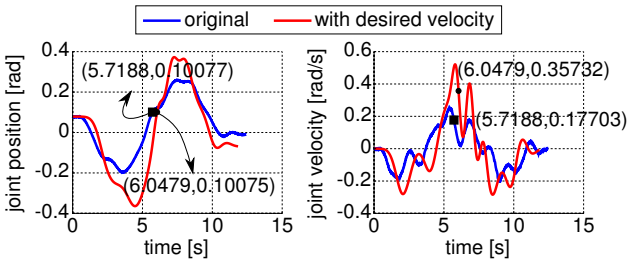


Figure 4.5.: Optimizing multiple phase parameters and amplitude to achieve double the velocity of an original movement when the 1^{st} joint angle crosses the value $0.1\,\text{rad}$ for the first time. Assuming the desired trajectory could be exactly tracked by the robot.

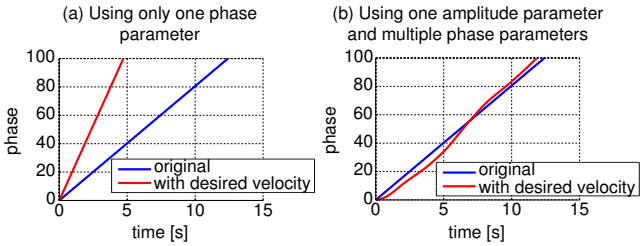


Figure 4.6.: Phase functions before and after optimization. (a) Phase functions of the form $z(t) = \alpha t$. (b) Phase functions with multiple phase parameters.

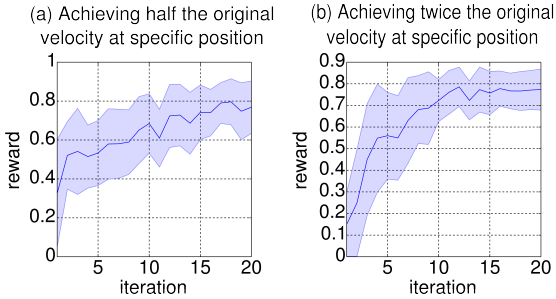


Figure 4.7.: Iteration of reinforcement learning algorithm versus expected reward with mean and standard deviation. (a) Evolution of the expected reward by trying to achieve half the original velocity at position $0.1\,\text{rad}$. (b) Evolution of expected reward by trying to achieve twice the original velocity at position $0.1\,\text{rad}$.

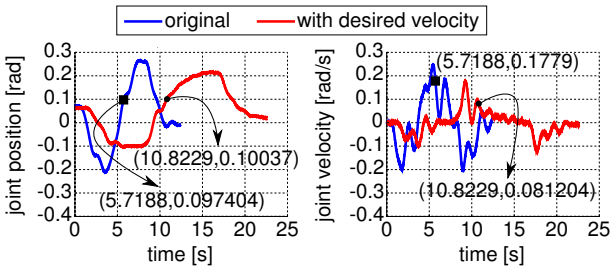


Figure 4.8.: Optimizing phase parameters and amplitude to achieve half the velocity of an original movement when the 1^{st} joint angle crosses the value $0.1\,\text{rad}$ for the first time. In this case, trajectories executed by the real robot have been evaluated.

Fig. 4.5 depicts the results of optimizing the phase parameters and the amplitude parameter after 100 iterations with 50 samples $\boldsymbol{\theta}_i$ each. The optimized trajectory has shape similar to the original one, except for the change in its amplitude. Velocities are not much higher than original ones far from the position of interest. Fig. 4.6 shows the phase functions before and after optimization with one and multiple phase parameters.

4.5.2 Optimizing Trajectories Executed by the Robot

In these experiments, the same Gaussian basis functions for shape and phase were used as in Section 4.5.1. The upper-level policy $\mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\right)$ was initialized with the same mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}^0$ as in Section 4.5.1. The initial covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^0$ was again a diagonal matrix and its diagonal was defined by the vector

$$\left[\sigma_{\alpha_2}^2 = 10, \cdots, \sigma_{\alpha_9}^2 = 10, \sigma_T^2 = 50000, \sigma_\lambda^2 = 0.25\right]^\top.$$

In a first experiment, the phase parameters were optimized to reach the half of the original velocity when the 1^{st} joint crosses position $0.1\,\text{rad}$ for the first time. In a second experiment, those same parameters were optimized to reach the double of the original velocity when the 1^{st} joint crosses this same specific position for the first time. In both experiments, there were 20 reinforcement learning iterations with 30 samples $\boldsymbol{\theta}_i$ each[3]. Fig. 4.7 shows how the expected reward changed with the number of iterations of the reinforcement learning algorithm. Figs. 4.8 and 4.9 show the trajectories found by the reinforcement learning algorithm that achieve the desired velocity in comparison to the original trajectories.

[3] The experiments with the real robot were time-consuming, since the robot took approximately 32 seconds on average to position the golf club and perform the putt swing. This motivated the choice of running 20 iterations with 30 samples $\boldsymbol{\theta}_i$ each.

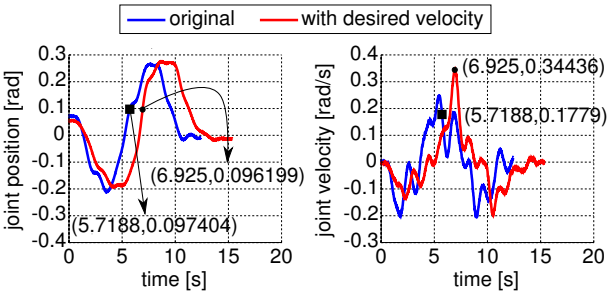


Figure 4.9.: Optimizing phase parameters and amplitude to achieve double the velocity of an original movement when the 1^{st} joint angle crosses the value $0.1\,\text{rad}$ for the first time. In this case, trajectories executed by the real robot have been evaluated.

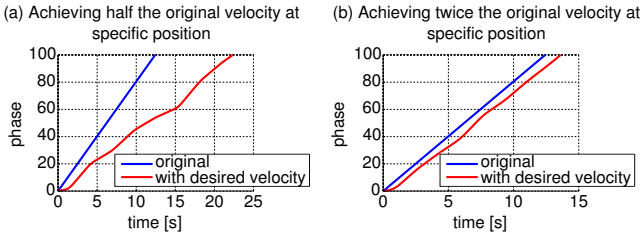


Figure 4.10.: Phase functions before and after optimization. The blue curve represents the original phase function, which was defined with constant slope. (a) The red curve represents the phase function after optimization to achieve half the original velocity when the 1^{st} joint reaches $0.1\,\text{rad}$ for the first time. (b) The red curve represents the phase function after optimization to achieve double the original velocity when the 1^{st} joint reaches $0.1\,\text{rad}$ for the first time.

In the solution for achieving half the original speed at a certain position, the overall duration of the movement is considerably longer than the original duration. The change in velocity is non-uniform, as can be clearly observed by the different slopes along the red curve in Fig. 4.10a.
In the solution for achieving twice the original speed at a certain position, the overall duration of the movement is slightly longer than the original duration. However, the movement accelerates in between, producing the desired velocity. This acceleration can be noticed by the slight change in slope along the red curve in Fig. 4.10b.

4.6 Epilogue

This chapter presented a movement primitive formulation with multiple phase parameters. Changes in these phase parameters allow for non-uniform acceleration or deceleration of a movement. We have shown that, using a reinforcement learning algorithm, it is possible to adapt the phase parameters and the amplitude of a movement demonstrated by a human in order to satisfy new velocity constraints. This was demonstrated in practice with experiments where an elastically actuated robot arm learned how to execute a golf putt swing, achieving new desired velocities at a specific position.
Experiments with more sophisticated reward functions might be performed with the real robot. For example, the reward function could favor energy-efficient movements. Our phase function with multiple parameters might play an important role in this case, since it allows for non-uniform changes in the speed of execution. Unnecessary accelerations or decelerations could then be avoided.
In the future, we will use a camera to detect the ball and a reward will be assigned to the movement of the robot depending on the final distance between the ball and the hole. Furthermore, we will evaluate the applicability of our movement primitive formulation to other tasks such as throwing or catching an object, moving objects around in an energy-efficient and safe way, etc.
So far in our experiments, only a small change in the shape of the movement through an amplitude parameter has been allowed. In a future work, we intend to allow for more general changes in shape alongside changes in phase in order to let the robot adapt to new constraints in space as well.

5 Assisting Movement Training and Execution with Visual and Haptic Feedback

In the practice of motor skills in general, errors in the execution of movements may go unnoticed when a human instructor is not available. In this case, a computer system or robotic device able to detect movement errors and propose corrections would be of great help. This chapter addresses the problem of how to detect such execution errors and how to provide feedback to the human to correct his/her motor skill using a general, principled methodology based on imitation learning. The core idea is to compare the observed skill with a probabilistic model learned from expert demonstrations. The intensity of the feedback is regulated by the likelihood of the model given the observed skill. Based on demonstrations, our system can, for example, detect errors in the writing of Japanese characters with multiple strokes. Moreover, by using a haptic device, the Haption Virtuose 6D, we demonstrate a method to generate haptic feedback based on a distribution over trajectories, which could be used as an auxiliary means of communication between an instructor and an apprentice. Additionally, given a performance measurement, the haptic device can help the human discover and perform better movements to solve a given task. In this case, the human first tries a few times to solve the task without assistance. Our framework, in turn, uses a reinforcement learning algorithm to compute haptic feedback, which guides the human towards better solutions.

5.1 Prologue

In the absence of an instructor, errors in the execution of movements by a person trying to learn a new motor skill, such as calligraphy, for example, may go unnoticed. To counter this problem, we propose recording demonstrations of a motor skill provided by an instructor and processing them such that someone practicing that motor skill in the absence of the instructor can have the correctness of his/her trials automatically assessed and receive feedback based on the demonstrations.

More precisely, our system aligns demonstrated trajectories in space and time and computes a probability distribution over them. Often, demonstrations may have been executed at different speeds. In order to extract the underlying shape of the movement from multiple trajectories, it is thus necessary to time-align these trajectories. In some cases, such as writing characters, the scale and the absolute position of the movements are not as relevant as their shape, justifying the necessity of addressing space-alignment in our framework as well.

When a new trajectory is executed, our system aligns the observations in space and time with the post-processed demonstrations and computes the probability of each of the positions of this new trajectory under the distribution over the demonstrations. The computed probabilities provide a way of assessing the correctness of each position of the new trajectory.

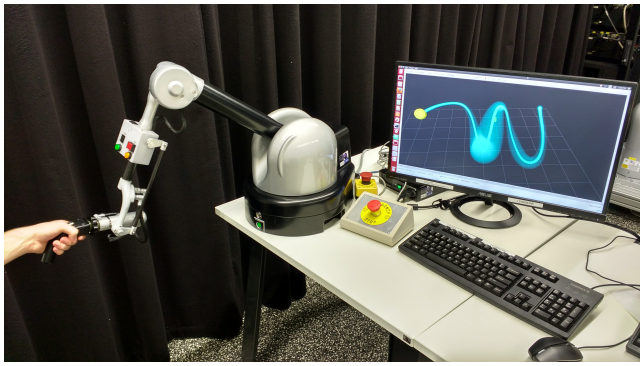


Figure 5.1.: Human manipulating a haptic device, the Haption Virtuose 6D. In our experiments, the haptic device assists the movements of the human by providing force feedback which is inversely proportional to the standard deviation of a distribution over trajectories (example is shown on the computer screen).

Based on this assessment, our system can generate visual or haptic feedback. We demonstrate the generation of visual feedback with the task of assisting the practice of writing Japanese characters on a monitor with a computer mouse. The generation of haptic feedback is demonstrated in an experiment with a haptic device, the Haption Virtuose 6D (see Figure 5.1). Our system gives haptic feedback to the user in the form of forces that constrain his/her movements when manipulating the haptic device, which can be seen as a form of guiding virtual fixtures [7]. The produced force is perpendicular to the mean trajectory of the distribution and its intensity is inversely proportional to the standard deviation along the distribution, as detailed in Section 5.4.

There are situations where the initial demonstrations are not enough to successfully accomplish a task, but it is possible to define performance measurements accounting for certain objectives. Examples of such a situation could be found in a

teleoperation task where the user perception and motor capabilities do not enable him/her to succeed. Such a task could be for instance telemanipulating a robot arm to move an object from a start position to an end position while avoiding obstacles. In such a task, a user can easily hit obstacles or fail to reach objects of interest. However, it may be possible to define performance measurements based on the positions of objects in the environment of the teleoperated robot. These positions could be computed from information provided by sensors in that environment. The framework presented in this chapter deals with these situations by applying reinforcement learning to adapt the original distribution over trajectories. The adapted distribution is then used to guide the user towards a better solution to the task.
In general, the problem of finding a distribution over trajectories that avoid obstacles and pass through positions of interest involves multiple optimization subproblems. Tuning the hyperparameters of the reward function to satisfy all the objectives may be time-consuming and may not produce the desired results. For this reason, our proposed framework includes a novel reinforcement learning algorithm that makes use of a parametric representation of trajectories and identifies how relevant each policy parameter is to each of the objectives of the task. By identifying how relevant each policy parameter is to each objective, it is possible to achieve effective policies with simpler reward functions, one for each objective, instead of a single reward function with different user-defined weights for each objective. Moreover, it is possible to optimize each objective sequentially, exploring different values of the parameters that matter for that objective and preserving the uncertainty about the other parameters.
In summary, this chapter presents a new framework to assist humans in training and executing movements by providing visual and haptic feedback to the user. This feedback can be given based on a probability distribution over expert demonstrations or based on an optimized distribution learned from a few non-expert demonstrations and performance criteria. By including methods for time and space-alignment of trajectories, this framework can potentially be applied to a large range of motor skills as long as the shape of the movement is critical, not its speed. In this work, our framework has been applied to the learning of Japanese characters and to teleoperation. As a secondary contribution, this chapter presents a novel reinforcement learning algorithm for problems involving multiple objectives, which are often encountered in teleoperation scenarios.

5.2 Related Work

This section primarily describes related work on techniques to assess the correctness of human motion and provide feedback to the user. It briefly introduces related work on the required components used for modeling the human demonstrations.

5.2.1 Human Motion Assessment and Feedback to the User

With similar goals as in our work, [3] presented a method to teach users how to write characters using a haptic interface. In their method, characters are modeled with Hidden Markov Models (HMMs) with discrete hidden states and discrete observations. The system recognizes online what character the user intends to write and applies a proportional derivative (PD) controller with fixed gains to restrict the user to move along the trajectory that corresponds to the recognized character. Differently, in our work, the gains of the haptic device are adapted as a function of the user's deviation with respect to the model learned from expert demonstrations or through reinforcement learning. Adaptive gains allow for practicing motor skills with multiple correct possibilities of execution, in case there is not a single correct trajectory. Also, it allows for regulating the stiffness of the robot to impose different levels of precision at different parts of the movement.
[32] proposed a "multilayer learning architecture with incremental self-organizing networks" to give the user real-time visual feedback during the execution of movements, e.g. powerlifting exercises. In our work, we have not addressed real-time visual feedback so far, although we do address real-time haptic feedback. On the other hand, our framework can deal with movements with different absolute positions and scales when producing visual feedback. By disabling this preprocessing, it would be possible to generate real-time visual feedback as well.
[31] presented a workflow to detect anomalies in weight training exercises. In their work, movement repetitions are segmented based on the acceleration along an axis in space. A probability distribution over a number of time-aligned repetitions is built. Then, based on this distribution, movement segments can be deemed correct or incorrect. Our approach focuses rather on correcting movements with respect to their shape or position in space, not on correcting acceleration patterns.
A variable impedance controller based on an estimation of the stiffness of the human arm was proposed by [4]. This controller enabled a robot to assist humans in calligraphic tasks. In the cited work, the tracked trajectories were not learned from demonstrations.
Our work is in line with approaches that aim to assist in human learning with demonstrations. [73], for instance, used probabilistic virtual guides learned from demonstrations to help humans manipulate a robot arm. In another related work, [74] presented a system that learns from demonstrations how to assist humans using a smart wheelchair.

Visual, auditory and haptic feedback modalities have been successfully used for motor learning in the fields of sport and rehabilitation [33]. Our method to provide visual feedback to the user, detailed in Section 5.3.4, is, for instance, similar in principle to bandwidth feedback. This sort of feedback means that the user only receives feedback when the movement error exceeds a certain threshold and it has been shown to be effective in rehabilitation [8]. The work here presented relates and can potentially complement previous research on bandwidth feedback in the sense that our threshold is not constant, but depends on a probability distribution over trajectories. Our approach may find applications in tasks where it is desirable to give the user more freedom of movement around a certain position and less freedom around a different position or where multiple variations of movements are considered correct.

[9] have demonstrated that maximum-likelihood estimation describes the way humans combine visual and haptic perception. The estimation of a certain environmental property that results from the combination of visual and haptic stimuli presents a lower variance than estimations based only on visual or haptic stimuli. When the visual stimulus is noise-free, users tend to rely more on vision to perform their estimation. On the other hand, when the visual stimulus is noisy, users tend to rely more on haptics. Therefore, users may profit from multimodal feedback to learn a new motor skill. In our experimental section, we provide haptic feedback to users to help them perform a teleoperation task in a virtual environment. The findings in [9] indicate that haptic feedback also helps users perceive some aspects of the task that they could not perceive only from visual stimuli, which could help them learn how to better solve the task without assistance next time. The work here presented offers an algorithmic solution to the acquisition of policies and control of a robotic device that could be applied to help humans learn and retain motor skills.

In contrast to most of the work on haptic feedback for human motor learning, our method modulates the stiffness of the haptic device according to demonstrations and uses reinforcement learning to improve upon the demonstrated movements. Those features may be interesting as a means of communication between an expert and an apprentice or patient and to enable improvement of initial demonstrations.

5.2.2 Learning and Adapting Models from Demonstrations

An essential component of this work is to construct a model from expert demonstrations, which is then queried at runtime to evaluate the performance of the user. One recurrent issue when building models from demonstration is the problem of handling the variability of phases (i.e. the speed of the execution) of different movements. [75] proposed the Continuous Profile Model (CPM), which can align multiple continuous time series. It assumes that each continuous time series is a non-uniformly subsampled, noisy and locally rescaled version of a single latent trace. The model is similar to a Hidden Markov Model (HMM). The hidden states encode the corresponding time step of the latent trace and a rescaling factor. The CPM has been successfully applied to align speech data and data sets from an experimental biology laboratory.

[44] augmented the model of [75] by additionally learning the dynamics of the controlled system in the vicinity of the intended trajectory. With this modification, their model generates an ideal trajectory that not only is similar to the demonstrations but also obeys the system's dynamics. Moreover, differently from [75], their algorithm to time-align the demonstrations and to determine an ideal trajectory relies both on an EM algorithm and on Dynamic Time Warping [40]. With this approach, they were able to achieve autonomous helicopter aerobatics after training with sub-optimal human expert demonstrations.

The same method was used by [45] to extract an ideal trajectory from multiple demonstrations. The demonstrations were, in this case, movements of a surgical robot operated by a human expert.

Similarly to [44,45], our system uses Dynamic Time Warping (DTW) to time-align trajectories. While DTW usually aligns pairs of temporal sequences, in Section 5.3.2 we present a solution for aligning multiple trajectories. An alternative solution was presented by[76], however, it suffers from scalability issues because distances need to be computed between every point of every temporal sequence.

Differences in the scale and shape of movements must also be addressed to account for the variability in human demonstrations. In practice, for tasks such as writing, we want our system to be invariant to the scale of the movements of different demonstrations. The analysis of the difference between shapes is usually addressed by Procrustes Analysis [77]. The output of this analysis is the affine transformation that maps one of the inputs to best match the other input, while the residual is quantified as the effective distance (deformation) between the shapes. As the analysis consists of computing such transformations in relation to the centroid, Procrustes Analysis provides a global, average assessment and has found applications in tasks of trajectory and transfer learning [78–80] and manipulation [81]. While this seems the most natural solution to our problem of aligning shapes, we noticed that it is not suitable for detecting anomalies. In fact, in the writing task, we are interested in finding the "outliers" that can be indicated to the human as erroneous strokes. However, Procrustes Analysis aligns the shapes globally such that the positions of the centroids are inappropriately biased towards such outliers. In Sections 5.3.1.1 and 5.3.1.2 we describe our own alignment method that is suited for detecting particular errors with the introduction of a few heuristics.

5.3 Processing Demonstrations and Assessing the Correctness of Observed Trajectories

Assuming the availability of expert demonstrations, the workflow of our proposed method is the following: First, the expert demonstrations are aligned in space and time and a probability distribution over these demonstrations is computed. Afterward, a user tries to perform the motor task. The movements of the user are also aligned in space and time with the demonstrations. Based on the probability distribution over the demonstrations, our system highlights which parts of the user's movements need improvement. A way of translating a distribution over trajectories into haptic feedback is presented later in Section 5.4. A novel reinforcement learning algorithm to help the user achieve good movements according to certain performance criteria without good demonstrations available is presented in Section 5.5.

5.3.1 Rescaling and Repositioning

In assessing the correctness of individual executions of a motor skill, it is often not important what the absolute position of the sequence of movements is, e.g. in weightlifting or gymnastics. In some situations, it is also not of crucial importance what the scale of the movements is as long as they keep their relative proportions, e.g. in drawing or calligraphy. Therefore, our system rescales all trajectories, both the ones demonstrated by a human expert and the ones performed by a user practicing a motor skill. Moreover, all trajectories are repositioned in such a way that the first position of the reference stroke is at the origin of the coordinate system. In practice, each stroke composing a motor skill is used once as the reference for rescaling and repositioning. For each reference stroke, a different score and visual feedback are computed. The best score and the respective feedback are presented to the user. This procedure enables our algorithm to present meaningful feedback to the user regardless of the location of his/her errors. In this section, our method for rescaling and repositioning is explained for two dimensions (x and y) and exemplified with the task of writing Japanese characters. This method can nevertheless be extended in a straightforward manner for more than two dimensions.

5.3.1.1 Rescaling

First, the system computes

$$\Delta x_{\text{ref}} = \max_t x_{\text{ref}}(t) - \min_t x_{\text{ref}}(t), \tag{5.1}$$

$$\Delta y_{\text{ref}} = \max_t y_{\text{ref}}(t) - \min_t y_{\text{ref}}(t), \tag{5.2}$$

where t indexes each time step, $\max_t x_{\text{ref}}(t)$ is the maximum x coordinate of the reference stroke, $\min_t x_{\text{ref}}(t)$ is the minimum x coordinate of the reference stroke, and similarly for $\max_t y_{\text{ref}}(t)$ and $\min_t y_{\text{ref}}(t)$.
Subsequently, a rescaling factor α is given by

$$\alpha = \begin{cases} \frac{1}{\Delta x_{\text{ref}}} & \text{if } \Delta x_{\text{ref}} \geq \Delta y_{\text{ref}}, \\ \frac{1}{\Delta y_{\text{ref}}} & \text{otherwise.} \end{cases} \tag{5.3}$$

The characters are written on a square window with side equal to 1. The rescaling factor α expresses the ratio between the constant 1 and the width Δx_{ref} or height Δy_{ref} of the reference stroke. If $\Delta x_{\text{ref}} \geq \Delta y_{\text{ref}}$, the width is used to compute α. Otherwise, the height is used. Some strokes are much larger in width than in height or vice versa. Therefore, this way of computing the rescaling factor selects the width or the height of the reference stroke according to which one will lead to the smallest amount of rescaling. For example, the characters depicted in Figure 5.2(a) will be rescaled according to the width of the first stroke of each of them respectively, resulting in characters whose first stroke has width equal to 1.
The rescaling factor can also be written as

$$\alpha = \frac{x_{i,\text{rescaled}}(t) - \min_{\{j,k\}} x_j(k)}{x_i(t) - \min_{\{j,k\}} x_j(k)} = \frac{y_{i,\text{rescaled}}(t) - \min_{\{j,k\}} y_j(k)}{y_i(t) - \min_{\{j,k\}} y_j(k)}, \tag{5.4}$$

where both t and k are time step indexes, while the indexes i and j represent the strokes of a character. Here, $x_{i,\text{rescaled}}(t) - \min_{\{j,k\}} x_j(k)$ is the difference between the x coordinate at time step t of stroke i after rescaling and the minimum x coordinate of the character. The term $x_i(t) - \min_{\{j,k\}} x_j(k)$ represents the corresponding difference before rescaling. Equation (5.4) also includes similar terms for the y coordinates. Therefore, after rescaling, the difference between the x coordinate of the position at time step t and the minimum x coordinate is α times this difference before rescaling, and similarly for the y coordinate. Thus this rescaling keeps the proportion between the width and the height of the character.

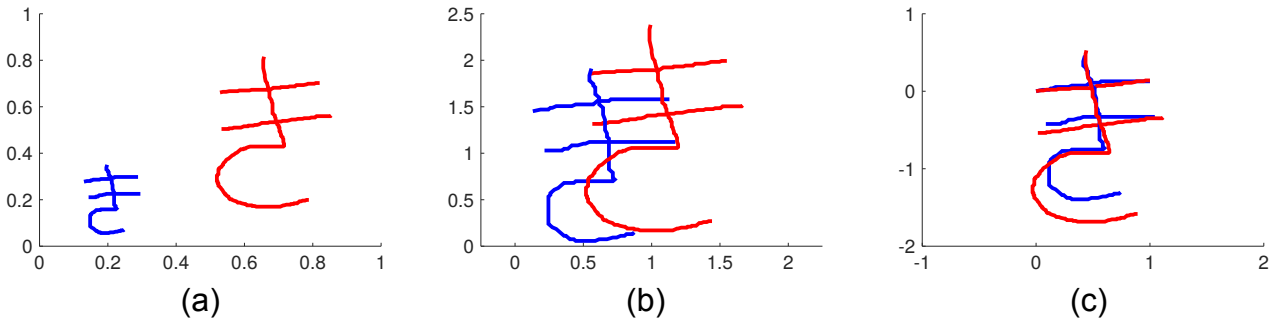


Figure 5.2.: Rescaling and repositioning different executions of a motor skill. In this example, the motor skill is writing a Japanese character. **(a)** Two demonstrations of a Japanese character. **(b)** After rescaling both characters. **(c)** After repositioning the characters such that the first position of the first stroke is $(x = 0, y = 0)$. The first stroke is the reference stroke in this case.

Rearranging the terms of (5.4) leads to

$$x_{i,\text{rescaled}}(t) = \min_{\{j,k\}} x_j(k) + \left(x_i(t) - \min_{\{j,k\}} x_j(k)\right)\alpha, \tag{5.5}$$

$$y_{i,\text{rescaled}}(t) = \min_{\{j,k\}} y_j(k) + \left(y_i(t) - \min_{\{j,k\}} y_j(k)\right)\alpha, \tag{5.6}$$

which is how the coordinates of the rescaled version of a character are computed. Figure 5.2(a) shows two demonstrations of the same character and Figure 5.2(b) shows the result of rescaling these characters.

5.3.1.2 Repositioning

In order to reposition a character such that the first position of the reference stroke is $(x = 0, y = 0)$, our system simply computes

$$x_{i,\text{repositioned}}(t) = x_i(t) - x_{\text{ref}}(t = 1), \tag{5.7}$$

$$y_{i,\text{repositioned}}(t) = y_i(t) - y_{\text{ref}}(t = 1), \tag{5.8}$$

where $x_i(t)$ and $y_i(t)$ are the original coordinates of stroke i at time step t, $x_{i,\text{repositioned}}(t)$ and $y_{i,\text{repositioned}}(t)$ are the coordinates of stroke i at time step t of the character after repositioning, $x_{\text{ref}}(t = 1)$ and $y_{\text{ref}}(t = 1)$ are the coordinates of the reference stroke at the first time step. Figure 5.2(c) shows two demonstrations of the same character after rescaling and repositioning.

5.3.2 Time Alignment

The time alignment of all the demonstrations and of the user's movements is achieved in our system by using Dynamic Time Warping [40]. Each stroke of an execution of a motor skill is time-aligned with respect to the corresponding stroke of other executions of that same motor skill.

Suppose two corresponding strokes need to be time-aligned. Let us represent these strokes by $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$, which are sequences of Cartesian coordinates from time step $t = 1$ until time step $t = T_1$ and $t = T_2$, respectively. Here, T_1 and T_2 represent the last time step of $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$, respectively.

First, the Euclidean distance $\boldsymbol{D}(i, j)$ between position at $t = i$ of $\boldsymbol{\tau_1}$ and position at $t = j$ of $\boldsymbol{\tau_2}$ is computed for all time steps of both strokes, i.e.

$$\boldsymbol{D}(i, j) = \|\boldsymbol{\tau}_1(i) - \boldsymbol{\tau}_2(j)\|, \tag{5.9}$$
$$\forall i \in \{1, 2, \cdots, T_1\}, \forall j \in \{1, 2, \cdots, T_2\}.$$

Algorithm 2 Path Search

```
1: procedure PATH(T_1, T_2, A)
2:     k ← 1
3:     i ← T_1
4:     j ← T_2
5:     p(k) ← (i, j)
6:     while i ≠ 1 or j ≠ 1 do
7:         if i = 1 then
8:             j ← j − 1
9:         else if j = 1 then
10:            i ← i − 1
11:        else
12:            if A(i−1, j) = min{A(i−1, j), A(i−1, j−1), A(i, j−1)} then
13:                i ← i − 1
14:            else if A(i, j−1) = min{A(i−1, j), A(i−1, j−1), A(i, j−1)} then
15:                j ← j − 1
16:            else
17:                i ← i − 1
18:                j ← j − 1
19:            end if
20:        end if
21:        k ← k + 1
22:        p(k) ← (i, j)
23:    end while
24:    return p
25: end procedure
```

Subsequently, assuming that the first position of τ_1 corresponds to the first position of τ_2, the accumulated cost $\boldsymbol{A}(i,j)$ of associating $\tau_1(i)$ with $\tau_2(j)$ is computed according to

$$\boldsymbol{A}(1,1) = \boldsymbol{D}(1,1), \tag{5.10}$$

$$\boldsymbol{A}(i,1) = \boldsymbol{D}(i,1) + \boldsymbol{A}(i-1,1), \tag{5.11}$$

$$\boldsymbol{A}(1,j) = \boldsymbol{D}(1,j) + \boldsymbol{A}(1,j-1), \tag{5.12}$$

$$\boldsymbol{A}(i,j) = \boldsymbol{D}(i,j) + \min\left\{\boldsymbol{A}(i-1,j), \boldsymbol{A}(i-1,j-1), \boldsymbol{A}(i,j-1)\right\}. \tag{5.13}$$

Algorithm 3 Warping for a Pair of Trajectories

```
1: procedure PAIRWARP(p, τ_1, τ_2)
2:     t ← 0
3:     for k = p.Length → 1 do
4:         t ← t + 1
5:         (i, j) ← p(k)
6:         τ'_1(t) ← τ_1(i)
7:         τ'_2(t) ← τ_2(j)
8:     end for
9:     return τ'_1, τ'_2
10: end procedure
```

Algorithm 4 Warping for Multiple Trajectories

```
1: procedure MULTIPLEWARP(τ_1, τ_2, ··· , τ_n)
2:     for l = 1 → n − 1 do
3:         (p, τ_l, τ_{l+1}) ← DTW(τ_l, τ_{l+1})
4:         for m = 1 → l − 1 do
5:             t ← 0
6:             for k = p.Length → 1 do
7:                 t ← t + 1
8:                 (i, j) ← p(k)
9:                 τ_m(t) ← τ_m(i)
10:            end for
11:        end for
12:    end for
13:    return τ_1, τ_2, ··· , τ_n
14: end procedure
```

Once the matrix of accumulated costs $\boldsymbol{A}$ has been determined, a path $\boldsymbol{p}$ can be computed that indicates how each trajectory should progress in time such that the minimum total cost is achieved. This path is computed backward in time in a dynamic programming fashion, as detailed in Algorithm 2.

The time-warped versions of trajectories τ_1 and τ_2, denoted by τ_1' and τ_2', are computed with Algorithm 3.
Algorithms 2 and 3 represent a common form of DTW which aligns pairs of temporal sequences. Algorithm 4 shows our proposed extension of DTW for time-aligning multiple temporal sequences. It works as follows: Trajectories τ_1 and τ_2 are time-aligned with DTW, resulting in τ_1' and τ_2'. Then τ_2' and τ_3 are time-aligned. Subsequently, the same warping applied to τ_2' is also applied to τ_1'. The algorithm proceeds like that until τ_n, always warping previous trajectories as well. For n trajectories, DTW needs to be computed $n-1$ times and the computation of the distance matrix $\boldsymbol{D}$ remains the same as in the original DTW. Figure 5.3 exemplifies the time-alignment of multiple trajectories.

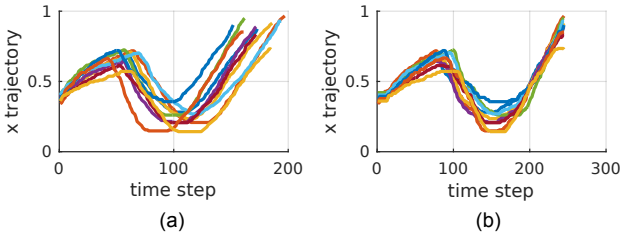


Figure 5.3.: x trajectories of corresponding strokes of multiple instances of a Japanese character. **(a)** Before time alignment. **(b)** After time alignment using DTW and our extension to deal with multiple trajectories.

5.3.3 Distribution over Trajectories

In order to create a distribution over trajectories, we use the framework of Probabilistic Movement Primitives [35]. Probabilistic Movement Primitives (ProMPs) allow for representing each trajectory with a relatively small number of parameters. A distribution over trajectories can then be computed by integrating out those parameters.
More precisely, in this framework, each trajectory $\boldsymbol{\tau}$ with a certain duration T is approximated by a weighted sum of N normalized Gaussian basis functions evenly distributed along the time axis. This approximation can be represented by

$$\boldsymbol{\tau} = \boldsymbol{\Psi}\boldsymbol{w} + \boldsymbol{\epsilon}, \tag{5.14}$$

where $\boldsymbol{w}$ is a weight vector, $\boldsymbol{\epsilon}$ is a zero-mean i.i.d. Gaussian noise, i.e. $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_{TxT}\right)$, and

$$\boldsymbol{\Psi} = \begin{bmatrix} \psi_1(1) & \psi_2(1) & \cdots & \psi_N(1) \\ \psi_1(2) & \psi_2(2) & \cdots & \psi_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(T) & \psi_2(T) & \cdots & \psi_N(T) \end{bmatrix}. \tag{5.15}$$

A term $\psi_i(t)$ in this matrix represents the normalized Gaussian basis function with index i evaluated at time step t.
Given a trajectory $\boldsymbol{\tau}$, a pre-defined matrix of basis functions $\boldsymbol{\Psi}$ and a regularizing factor λ, the weight vector $\boldsymbol{w}$ can be computed with linear ridge regression according to

$$\boldsymbol{w} = \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \lambda \boldsymbol{I}_{NxN}\right)^{-1} \boldsymbol{\Psi}^\top \boldsymbol{\tau}. \tag{5.16}$$

Once the weight vectors $\boldsymbol{w}$ corresponding to a set of trajectories $\boldsymbol{\tau}$ have been computed, a Gaussian distribution $\mathcal{N}\left(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right)$ over these vectors is determined using maximum likelihood estimation. The distribution over trajectories $\boldsymbol{\tau}$ can be expressed as the marginal distribution

$$p(\boldsymbol{\tau}) = \int p(\boldsymbol{\tau}|\boldsymbol{w})\, p(\boldsymbol{w})\, d\boldsymbol{w}, \tag{5.17}$$

where $p(\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right)$. Assuming that a Gaussian is a good approximation for the distribution over $\boldsymbol{w}$, this integral can be solved in closed-form, yielding

$$p(\boldsymbol{\tau}) = \mathcal{N}\left(\boldsymbol{\tau}|\boldsymbol{\mu}_\tau, \boldsymbol{\Sigma}_\tau\right), \tag{5.18}$$

with

$$\begin{aligned} \boldsymbol{\mu}_\tau &= \boldsymbol{\Psi}\boldsymbol{\mu}_w, \\ \boldsymbol{\Sigma}_\tau &= \sigma^2 \boldsymbol{I}_{TxT} + \boldsymbol{\Psi}\boldsymbol{\Sigma}_w\boldsymbol{\Psi}^\top. \end{aligned} \tag{5.19}$$

To deal with not only one stroke and a single degree of freedom (DoF) but with multiple strokes and multiple DoFs, one can think of $\boldsymbol{\tau}$ as a concatenation of trajectories. The matrix $\boldsymbol{\Psi}$ becomes, in this case, a block diagonal matrix and $\boldsymbol{w}$ a concatenation of weight vectors. For further details about this formulation, the interested reader is referred to our previous work [82] in which ProMPs were used to coordinate the movements of a human and a robot in collaborative scenarios.
The variance σ^2 defining the Gaussian noise $\boldsymbol{\epsilon}$ determines how sensitive our system is to deviations from the distribution over demonstrations because σ^2 directly influences the variance along this distribution, as expressed by (5.19). A small σ^2 results in assessing positions as incorrect more often, while a high σ^2 results in a less strict evaluation.

5.3.4 Assessing the Correctness of New Trajectories

The correctness of each position of a new trajectory is assessed by comparing the probability density function evaluated at that position with the probability density function evaluated at the corresponding position along the mean trajectory, which is considered by our system the best achievable trajectory, since it is the one with the highest probability under the Gaussian distribution over demonstrations.
First, the ratio

$$g(t) = \frac{p(\tau(t))}{p\left(\mu_\tau(t)\right)} \tag{5.20}$$

is computed for each time step t, where $p(\tau(t))$ is the probability of position $\tau(t)$ at time step t and $p\left(\mu_\tau(t)\right)$ is the probability of position $\mu_\tau(t)$ at time step t. Since the highest achievable value of the Gaussian probability density function at each time step is the one achieved by the mean trajectory, g is a function with values between 0 and 1.
Subsequently a score

$$s(g(t)) = \frac{\arctan((g(t)+a)\,b)}{2c} + 0.5 \tag{5.21}$$

for each time step t is computed, where

$$c = \arctan((1+a)\,b). \tag{5.22}$$

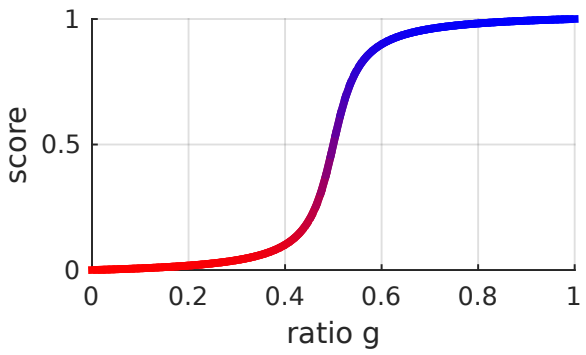


Figure 5.4.: Score function relating the ratio g between the probability of a certain position and the probability of the corresponding position along the mean trajectory. This function is determined by (5.21) and was designed to be 0 when $g = 0$, 1 when $g = 1$ and to monotonically increase with g. It is possible to change the steepness of this function by changing its hyperparameter b. The same color code as in this figure is used to give visual feedback to the user.

The score function s was designed with a few desired properties in mind. With $a = -0.5$, s is equal to 0 when the ratio g is equal to 0, it is 0.5 when g is 0.5 and it is 1 when g is 1. The score function s monotonically increases with g. Its steepness is regulated by the parameter b. We have been using $a = -0.5$ and $b = 25$. One could consider using other score functions, depending on the preferences of the users. The score function depicted in Figure 5.4 leads to a sharp distinction between right and wrong positions. One might prefer a more gradual distinction. In this work, we did not investigate what score function the users prefer nor whether certain score functions make the users learn faster. These considerations could be the subject of extensive user studies.

5.4 Method to Provide Haptic Feedback

Up to now, it has been solely discussed in this chapter how to provide offline visual feedback to the user assessing the correctness of his/her movements. Here, it is presented how our framework provides online haptic feedback to the user, guiding him/her towards correct movements.
The Haption Virtuose 6D can provide force feedback to the user by simulating a virtual object attached to its end-effector constituting a mass-spring-damper system. Given the mass and the inertia of the virtual object, the Virtuose API computes stiffness and damping coefficients that guarantee the stability of the system. The intensity of the force produced by this system can be rescaled by a factor denoted in this chapter by ζ.
In this work, we are interested in providing feedback to the user according to a probability distribution over trajectories, which is computed as in Section 5.3.3. If the standard deviation at a certain part of the distribution is high, the haptic device should become compliant in that region, while if the standard deviation is low, the haptic device should become stiff. The virtual object always lies along the mean trajectory of the distribution. The factor ζ can be derived from

$$\frac{\zeta - \zeta_{\min}}{\zeta_{\max} - \zeta_{\min}} = \frac{\sigma - \sigma_{\max}}{\sigma_{\min} - \sigma_{\max}}, \tag{5.23}$$

where $\zeta_{\min}$ and $\zeta_{\max}$ are respectively the minimum and the maximum force scaling factor. These values have been empirically defined in our experiments. The variable σ stands for the standard deviation that corresponds to the current position of the virtual object. The variables $\sigma_{\min}$ and $\sigma_{\max}$ stand for the minimum and maximum standard deviations of the distribution over trajectories. These values can be determined from a set of demonstrated trajectories. The underlying

assumption behind (5.23) is that the stiffness is the highest when the standard deviation is the minimum and the lowest when the standard deviation is the maximum. Moreover, we assume a linear dependence between $\zeta - \zeta_{\min}$ and $\sigma - \sigma_{\max}$. Rearranging (5.23), we get

$$\zeta = \zeta_{\min} + (\zeta_{\max} - \zeta_{\min})\left(\frac{\sigma - \sigma_{\max}}{\sigma_{\min} - \sigma_{\max}}\right). \tag{5.24}$$

The closest point along the mean trajectory that is not further away from the previous position of the virtual object than a certain threshold becomes the new position of the virtual object. This threshold is especially necessary when dealing with convoluted trajectories to avoid large sudden variations in the position of the virtual object.
In situations where there are no good demonstrations available, but there is a performance measurement of the trajectories, it is possible to use reinforcement learning to improve the distribution over trajectories. Such a situation could be found in a teleoperation scenario, where an optimization problem with multiple objectives may have to be solved, accounting for distances to via points, distances to obstacles and other performance measurements. In the next section, a novel reinforcement learning algorithm is presented to address such problems.

5.5 Relevance Weighted Policy Optimization

We are interested in enabling a haptic device to assist a human in a task also when good demonstrations are not available. As will be presented in Section 5.6.2, our particular task is to move an object in a virtual environment from a start position to an end position through a window in a wall. We have defined three objectives to determine the performance of solutions to this task: distance to the start position, distance to the center of the window and distance to the end position. An optimal policy for this task is a trajectory that begins at the start position, passes through the center of the window and reaches the end position. This problem can be decomposed into three subproblems w.r.t. which a policy parameter can be more or less relevant. Therefore, in this section, a new policy search method is explained, which identifies the relevance of each policy parameter to each subproblem in order to improve the learning of the global task. This method makes use of Reward-weighted Regression [71]. The basic idea of this method is to first find out how much each policy parameter influences each objective. Subsequently, this information is used to optimize the policy with respect to the objectives. In our particular application, the policy parameters are the elements of the weight vector $\boldsymbol{w}$ as in (5.14).

5.5.1 Learning Relevance Functions

Our approach to answering how much each policy parameter influences each objective consists of learning a relevance function f_o for each objective o. The argument of this function is an index identifying a policy parameter. In other words, a relevance function $f_o(n)$ evaluated for policy parameter indexed by n represents how relevant this parameter is to the objective indexed by o. In order to learn this function, in this chapter, it is assumed that a relevance function can be represented by a weighted sum of basis functions with lower bound 0 and upper bound 1 according to the expression

$$f_o(n) = \begin{cases} 0, & \text{if } \boldsymbol{\rho}^\top \boldsymbol{\phi}(n) \leq 0 \\ 1, & \text{if } \boldsymbol{\rho}^\top \boldsymbol{\phi}(n) \geq 1\,, \\ \boldsymbol{\rho}^\top \boldsymbol{\phi}(n), & \text{otherwise,} \end{cases} \tag{5.25}$$

where $\boldsymbol{\rho}$ is a vector of weights ρ_i for the basis functions ϕ_i and $\boldsymbol{\phi}(n) = [\phi_1(n), \phi_2(n), \cdots, \phi_I(n)]^\top$. It will become clear in the remainder of this section why the lower bound of a relevance function is 0 and its upper bound is 1.
The basis functions are

$$\phi_i(n) = \frac{1}{\exp(-k(n - m_i))}, \tag{5.26}$$

$$\phi_I = 1, \tag{5.27}$$

with $i \in \{1, 2, \cdots, I\}$, where I is the total number of basis functions for the relevance function, n is an index representing one of the policy parameters, k is a scalar determining steepness and m_i is a scalar determining the midpoint of the logistic basis function with index i.
These basis functions have been chosen because weighted combinations of them lead to reasonable relevance functions. For example, three relevance functions that can be constructed with the proposed basis functions are depicted in Fig. 5.5.

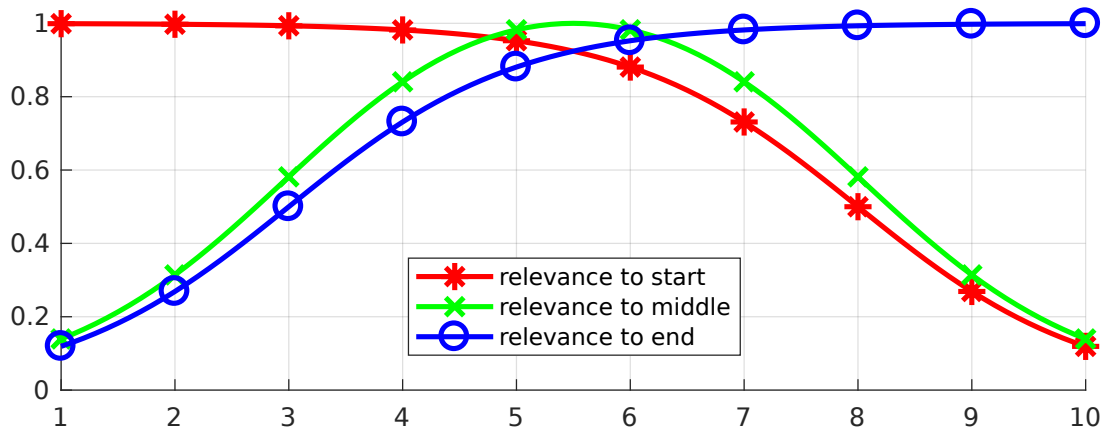


Figure 5.5.: Three examples of relevance functions. Let us assume that our goal is to optimize a certain movement with respect to an objective at the beginning of the movement, an objective in the middle and an objective in the end. Let us further assume that the movement to be optimized can be determined by 10 parameters and that the first parameters (close to 1) have higher influence over the beginning of the movement, while the last ones (close to 10) have higher influence over the end. The image depicts potentially suitable relevance functions for each of the objectives in this problem.

The depicted relevance functions determine how each of the parameters determining a movement influences objectives at the beginning of the movement, in the middle or in the end. These relevance functions are

$$f_{\text{start}}(n) = \phi_3(n) - \phi_2(n), \tag{5.28}$$

$$f_{\text{middle}}(n) = \frac{1}{\max_n(\phi_1(n) - \phi_2(n))}\phi_1(n) - \frac{1}{\max_n(\phi_1(n) - \phi_2(n))}\phi_2(n), \tag{5.29}$$

$$f_{\text{end}}(n) = \phi_1(n), \tag{5.30}$$

where the basis functions are

$$\phi_1(n) = \frac{1}{\exp(-(n-3))}, \tag{5.31}$$

$$\phi_2(n) = \frac{1}{\exp(-(n-8))}, \tag{5.32}$$

$$\phi_3(n) = 1. \tag{5.33}$$

In this framework, learning a relevance function with respect to a certain objective means finding a vector $\boldsymbol{\rho}$ that leads to high variability in the value of that objective and to low variability in the values of other objectives. How a relevance function influences the variability in the values of an objective will be made explicit in the following.
First, a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_\rho, \boldsymbol{\Sigma}_\rho)$ over $\boldsymbol{\rho}$ is initialized with a certain mean $\boldsymbol{\mu}_\rho$ and a certain covariance matrix $\boldsymbol{\Sigma}_\rho$. Subsequently, parameter vectors $\boldsymbol{\rho}$ are sampled from this distribution and, for each sample, the relevance function f_o is computed using (5.25).
Let us now assume that there is an initial Gaussian probability distribution $\mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ over the policy parameters $\boldsymbol{w}$. The mean $\boldsymbol{\mu}_w$ and the covariance matrix $\boldsymbol{\Sigma}_w$ can be computed from an initial set of demonstrations or determined by the user.
For each f_o computed with the sampled vectors $\boldsymbol{\rho}$, our algorithm generates samples of the policy parameters $\boldsymbol{w}$ from the distribution $\mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w{}^{f_o})$, where

$$\boldsymbol{\Sigma}_w{}^{f_o} = \begin{bmatrix} \sigma^2_{w_1} f_o(1) & 0 & \cdots & 0 \\ 0 & \sigma^2_{w_2} f_o(2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2_{w_N} f_o(N) \end{bmatrix} \tag{5.34}$$

and $\sigma^2_{w_n}$, $\forall n \in \{1, 2, \cdots, N\}$, are the variances in the diagonal of the matrix $\mathbf{\Sigma}_{\boldsymbol{w}}$. In other words, the policy parameters are sampled in such a way that their original variance is weighted with a relevance coefficient. The higher the relevance of a parameter, the larger the range of values for that parameter among the samples.
Each sampled vector of policy parameters $\boldsymbol{w}$ determines a policy with a corresponding value for each objective. In our teleoperation scenario, for example, each policy parameter vector $\boldsymbol{w}$ determines a trajectory, which has a certain distance to the start position, a certain distance to the center of the window and a certain distance to the end position. Given these objective values, our algorithm computes a reward function

$$R_{\boldsymbol{\rho},o} = \exp\left(\beta_{\text{relevance}}\left(\sigma_o - \sum_{i \neq o} \sigma_i\right)\right), \tag{5.35}$$

where σ_o is the standard deviation of the values for objective o and σ_i with $i \neq o$ is the standard deviation of the values for the other objectives. The scalar $\beta_{\text{relevance}}$ can be determined with line search.
Parameters $\boldsymbol{\rho}$ determining suitable relevance functions f_o result in higher reward $R_{\boldsymbol{\rho},o}$ because the range of values for the parameters that mainly affect objective o will be high, producing a high standard deviation of the values for that objective. Moreover, the range of values for the parameters that mainly affect the other objectives will be low, producing a low standard deviation of the values for the other objectives.
Finally, Reward-weighted Regression (RWR) is used to learn the relevance parameters $\boldsymbol{\rho}$. RWR is an iterative algorithm that finds the best Gaussian distribution over parameters of interest (in the particular case of optimizing the relevance functions, the parameters of interest are given by $\boldsymbol{\rho}$) to maximize the expected reward, given samples from the Gaussian distribution of the previous iteration. In order to do so, RWR solves the optimization problem

$$\{\boldsymbol{\mu}_{\boldsymbol{\rho}}^{k+1}, \mathbf{\Sigma}_{\boldsymbol{\rho}}^{k+1}\} = \underset{\{\boldsymbol{\mu}_{\boldsymbol{\rho}}, \mathbf{\Sigma}_{\boldsymbol{\rho}}\}}{\arg\max} \sum_{i=1}^{S} R_{\boldsymbol{\rho},o,i} \mathcal{N}\left(\boldsymbol{\rho}_i; \boldsymbol{\mu}_{\boldsymbol{\rho}}, \mathbf{\Sigma}_{\boldsymbol{\rho}}\right) \tag{5.36}$$

at each iteration, where S is the number of sampled parameter vectors $\boldsymbol{\rho}_i$ from the previous distribution $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\rho}}^k, \mathbf{\Sigma}_{\boldsymbol{\rho}}^k\right)$. The solution to this optimization problem is

$$\boldsymbol{\mu}_{\boldsymbol{\rho}}^{k+1} = \frac{\sum_{i=1}^{S} R_{\boldsymbol{\rho},o,i} \boldsymbol{\rho}_i}{\sum_{i=1}^{S} R_i}, \tag{5.37}$$

$$\mathbf{\Sigma}_{\boldsymbol{\rho}}^{k+1} = \frac{\sum_{i=1}^{S} R_{\boldsymbol{\rho},o,i} \left(\boldsymbol{\rho}_i - \boldsymbol{\mu}_{\boldsymbol{\rho}}^k\right)\left(\boldsymbol{\rho}_i - \boldsymbol{\mu}_{\boldsymbol{\rho}}^k\right)^\top}{\sum_{i=1}^{S} R_{\boldsymbol{\rho},o,i}}. \tag{5.38}$$

This procedure is repeated until convergence of $R_{\boldsymbol{\rho},o}$ has been reached to learn a relevance function f_o for each objective o. The parameters determining the relevance functions f_o are given by the vector $\boldsymbol{\mu}_{\boldsymbol{\rho}}$ computed in the last iteration. After this iterative procedure is finished, our algorithm computes $f_o(n) / \max_n f_o(n)$, $\forall n \in \{1, 2, \cdots, N\}$, and assigns this value to $f_o(n)$. This last step makes the maximum value of f_o be not less than 1 and helps the exploration in the policy optimization phase, which will be discussed in the next section. Algorithm 5 presents an informal description of the relevance learning algorithm.

5.5.2 Policy Optimization using Relevance Functions

Now that a relevance function for each objective has been learned, our algorithm uses this information to optimize a policy with respect to each objective. As in Section 5.5.1, it is assumed here that there is an initial Gaussian probability distribution $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{w}}, \mathbf{\Sigma}_{\boldsymbol{w}}\right)$ over the policy parameters $\boldsymbol{w}$.
For each objective o, our algorithm samples policy parameters $\boldsymbol{w}$ from the distribution $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{w}}, \mathbf{\Sigma}_{\boldsymbol{w}}{}^{f_o^*}\right)$, where

$$\mathbf{\Sigma}_{\boldsymbol{w}}{}^{f_o^*} = \begin{bmatrix} \sigma^2_{w_1} f_o^*(1) & 0 & \cdots & 0 \\ 0 & \sigma^2_{w_2} f_o^*(2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2_{w_N} f_o^*(N) \end{bmatrix} \tag{5.39}$$

and f_o^* is the learned relevance function with respect to the objective o. Therefore, the policy parameters $\boldsymbol{w}$ are sampled from a Gaussian distribution where the original variances $\sigma^2_{w_n}$ are weighted with the learned relevance function. This

Algorithm 5 Learning Relevance Functions

Inputs: mean $\boldsymbol{\mu}_{\boldsymbol{w}}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ of the policy parameter vectors $\boldsymbol{w}$
Initialize the mean $\boldsymbol{\mu}_{\boldsymbol{\rho}}$ and the covariance $\boldsymbol{\Sigma}_{\boldsymbol{\rho}}$ of the Gaussian distribution over the parameter vectors $\boldsymbol{\rho}$ that determine the relevance functions f_o
repeat
    Sample parameter vectors $\boldsymbol{\rho}$ from $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\rho}}, \boldsymbol{\Sigma}_{\boldsymbol{\rho}}\right)$
    for each sample vector $\boldsymbol{\rho}$ **do**
        for each objective o **do**
            Compute the relevance functions f_o (Equation 5.25)
            Compute matrix ${\boldsymbol{\Sigma}_{\boldsymbol{w}}}^{f_o}$ (Equation 5.34)
            Sample policy parameter vectors $\boldsymbol{w}$ from $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{w}}, {\boldsymbol{\Sigma}_{\boldsymbol{w}}}^{f_o}\right)$
            for each sample vector $\boldsymbol{w}$ **do**
                Compute value achieved by policy for objective o
            end for
            Compute standard deviation σ_o of the values achieved for o with the different samples
        end for
        for each objective o **do**
            Compute $R_{\boldsymbol{\rho},o}$ (Equation 5.35)
        end for
    end for
    Update $\boldsymbol{\mu}_{\boldsymbol{\rho}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\rho}}$ (Equations 5.37 and 5.38)
until convergence of the rewards $R_{\boldsymbol{\rho},o}$
$\boldsymbol{\rho}^* = \boldsymbol{\mu}_{\boldsymbol{\rho}}$
for each objective o **do**
    Compute f_o^* using $\boldsymbol{\rho}^*$ (Equation 5.25)
    Normalize f_o^* by computing $\frac{f_o^*(n)}{\max_n f_o^*(n)}$
end for
return the relevance functions f_o^*

procedure means that a larger range of values will be sampled for the policy parameters w_n that are more relevant to the objective o and a smaller range of values will be sampled for the policy parameters w_n that are less relevant to this objective.

For each sampled vector of policy parameters $\boldsymbol{w}_i$, the reward $R_{\boldsymbol{w},o,i}$ associated with the objective o is computed. These objectives and rewards depend on the problem. An objective might be for instance to achieve a certain goal position, in which case the reward could be a non-negative function monotonically decreasing with the distance to the goal position. In our particular teleoperation problem, the reward associated with the objective of being close to the start position is given by $R = \exp\left(-\beta_{\text{policy}} d_{\text{start}}\right)$, where d_{start} is the distance between the first position of the trajectory and the position where the trajectories should start.

Our algorithm uses once again RWR. This time, RWR is used to maximize the expected reward with respect to $\boldsymbol{\mu}_{\boldsymbol{w}}$ and ${\boldsymbol{\Sigma}_{\boldsymbol{w}}}^{f_o^*}$. This maximization is done iteratively according to

$$\{\boldsymbol{\mu}_{\boldsymbol{w}}^{k+1}, \boldsymbol{C}^{k+1}\} = \underset{\{\boldsymbol{\mu}_{\boldsymbol{w}}, {\boldsymbol{\Sigma}_{\boldsymbol{w}}}^{f_o^*}\}}{\arg\max} \sum_{i=1}^{S} R_{\boldsymbol{w},o,i} \mathcal{N}\left(\boldsymbol{w}_i; \boldsymbol{\mu}_{\boldsymbol{w}}, {\boldsymbol{\Sigma}_{\boldsymbol{w}}}^{f_o^*}\right), \tag{5.40}$$

where S is the number of sampled policy parameter vectors $\boldsymbol{w}_i$ from the previous distribution $\mathcal{N}\left(\boldsymbol{w}; \boldsymbol{\mu}_{\boldsymbol{w}}^k, {\boldsymbol{\Sigma}_{\boldsymbol{w}}}^{f_o^* k}\right)$.

The solution to (5.40) is given by

$$\boldsymbol{\mu}_{\boldsymbol{w}}^{k+1} = \frac{\sum_{i=1}^{S} R_{\boldsymbol{w},o,i} \boldsymbol{w}_i}{\sum_{i=1}^{S} R_{\boldsymbol{w},o,i}}, \tag{5.41}$$

$$\boldsymbol{C}^{k+1} = \frac{\sum_{i=1}^{S} R_{\boldsymbol{w},o,i} \left(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}}^k\right)\left(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}}^k\right)^\top}{\sum_{i=1}^{S} R_{\boldsymbol{w},o,i}}. \tag{5.42}$$

In each iteration, after computations (5.41) and (5.42), our algorithm updates the variances of each policy parameter $\sigma_{w_n}^2$ with

$$\sigma_{w_n,k+1}^2 = (1 - f_o(n))\, \sigma_{w_n,k}^2 + f_o(n)\, \boldsymbol{C}_{nn}^{k+1}, \tag{5.43}$$

where $\sigma^2_{w_n,k}$ is the previous variance of policy parameter w_n and $\boldsymbol{C}^{k+1}_{nn}$ is the n^{th} element along the main diagonal of the matrix $\boldsymbol{C}^{k+1}$. This equation has the effect of keeping the previous variance of the parameters that are less relevant to the objective o while updating the variance of the parameters that are more relevant to this objective. The algorithm then uses $\sigma^2_{w_n,k+1}$ to compute $\boldsymbol{\Sigma_w}^{f_o^*k+1}$ as in (5.39).
Finally, Equation 5.43 justifies the lower bound of 0 and the upper bound of 1 for the relevance function. The closer the relevance of policy parameter w_n is to 0, the closer the updated variance of this parameter is to the previous variance $\sigma^2_{w_n,k}$. The closer the relevance of policy parameter w_n is to 1, the closer the updated variance of this parameter is to $\boldsymbol{C}^{k+1}_{nn}$. In other words, the previous variance of irrelevant policy parameters is preserved, while the variance of relevant policy parameters is updated. Algorithm 6 presents an informal description of the algorithm for policy optimization using relevance functions.

Algorithm 6 Policy Optimization using Relevance Functions

Inputs: mean $\boldsymbol{\mu_w}$ and covariance $\boldsymbol{\Sigma_w}$ of the policy parameter vectors $\boldsymbol{w}$, learned relevance functions f_o^*
repeat
  for each objective o **do**
    Compute matrix $\boldsymbol{\Sigma_w}^{f_o^*}$ (Equation 5.39)
    Sample policy parameter vectors $\boldsymbol{w}$ from $\mathcal{N}\left(\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}^{f_o^*}\right)$
    for each sample vector $\boldsymbol{w}$ **do**
      Compute the reward $R_{\boldsymbol{w},o}$ of the policy with parameters given by vector $\boldsymbol{w}$ associated with objective o
    end for
    Update $\boldsymbol{\mu_w}$ and compute $\boldsymbol{C}$ (Equations 5.41 and 5.42)
    Update the variances of the policy parameters $\sigma^2_{w_n}$ (Equation 5.43)
  end for
until convergence of the rewards $R_{\boldsymbol{w},o}$
return the mean $\boldsymbol{\mu_w}$ and the variances $\sigma^2_{w_n}$

5.5.3 Example of Policy Optimization with Relevance Weighting

In order to make the proposed Relevance Weighted Policy Optimization algorithm more clear, we present an example using the 2D scenario depicted in Figure 5.6(a). This scenario is composed of a start position, a wall with a window and an end position. Given the initial trajectories depicted in Figure 5.6(a), the goal of our algorithm is to find a distribution over trajectories that begin at the start position, pass through the center of the window and reach the end position.
First, the algorithm aligns the initial trajectories in time and computes the parameters $\boldsymbol{w}$ for each of them using (5.16). Subsequently, the relevance functions for start position, center and end position are learned as in Section 5.5.1. An example of learned relevance functions is depicted in Figure 5.6(b). After learning the relevance functions, the algorithm uses the procedure explained in Section 5.5.2 to learn a policy that satisfies the three above-stated objectives. Figure 5.7 shows how the distribution over trajectories changes with the number of iterations of the algorithm. The distances to start, center and end positions decrease with the number of iterations and the return $\exp\left(-\beta_{\text{policy}}\left(d_{start} + d_{center} + d_{end}\right)\right)$ increases, as depicted by Figure 5.8. Here, β_{policy} is a parameter which can be determined with line search, d_{start} is the distance to the start, d_{center} is the distance to the center and d_{end} is the distance to the end.
Relevance Weighted Policy Optimization implements policy search for each objective sequentially. For each objective, the algorithm samples a larger range of values for the parameters that are more relevant to that objective, while sampling values close to the mean for the parameters that are less relevant. Subsequently, the algorithm optimizes the mean and the variances of the policy parameters given the samples. After optimization, the mean and the variance of the parameters that matter more to that objective are updated, while the mean and the variance of parameters that matter less remain similar to the previous distribution. The algorithm does not require defining a reward function with different weights for the different objectives, which can be time-consuming and ineffective. Moreover, at each iteration, when optimizing the distribution over policy parameters with respect to a certain objective, the algorithm does not accidentally find solutions that are good according to this objective, but bad according to the other objectives because only the mean and the variance of the parameters that matter change substantially. The mean and the variance of the other parameters remain close to the mean and the variance of the previous distribution.
Figure 5.9 exemplifies how the algorithm samples trajectories in the 2D problem. Figure 5.9(a) shows samples from the original distribution. Figure 5.9(b) shows samples of the first iteration of the algorithm right before optimizing for beginning at the start position. Figure 5.9(c) depicts the next step, still in the first iteration, after the first optimization for starting at the start position and before optimizing for passing through the center of the window. Finally, Figure 5.9(d) shows samples at the first iteration of the algorithm, right before optimizing for reaching the end position.

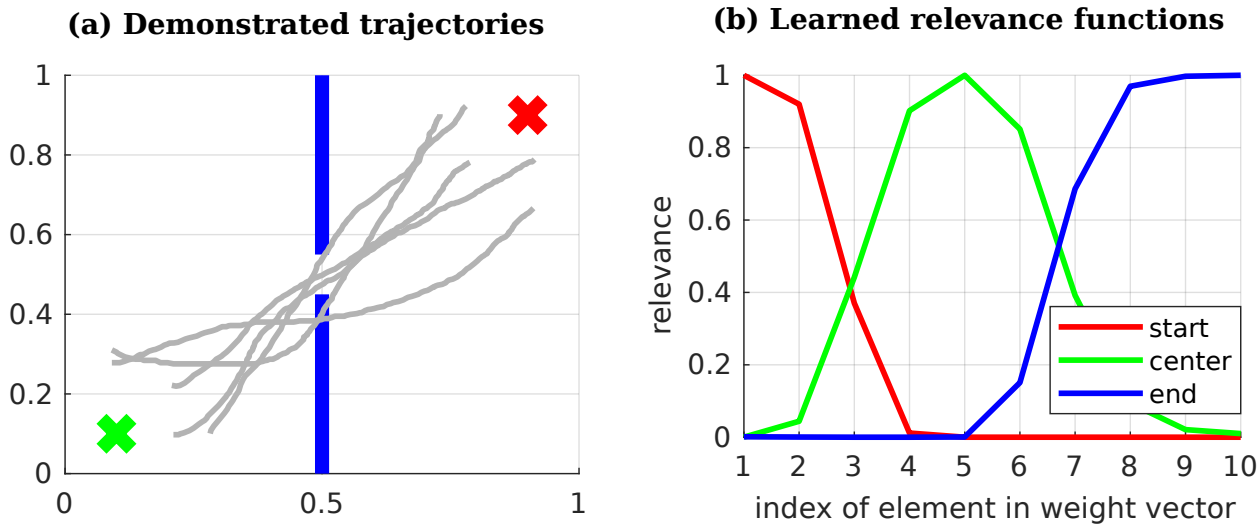


Figure 5.6.: **(a)** 2D problem used to explain the proposed Relevance Weighted Policy Optimization (RWPO) algorithm. The green x at the lower left corner of the image represents the start position. The blue lines in the middle represent a wall with a window in the center. The red x at the upper-right corner represents the end position. The goal of our algorithm is, given a few initial trajectories (depicted in light gray), to find a distribution over trajectories that begin at the start position, pass through the center of the window and reach the end position. **(b)** Learned relevance functions for the 2D problem. The learned relevance functions show that policy parameters close to w_1 are more important for beginning at the start position, policy parameters around w_5 are more important to pass through the center of the window and policy parameters close to w_{10} are more important to reach the end position.

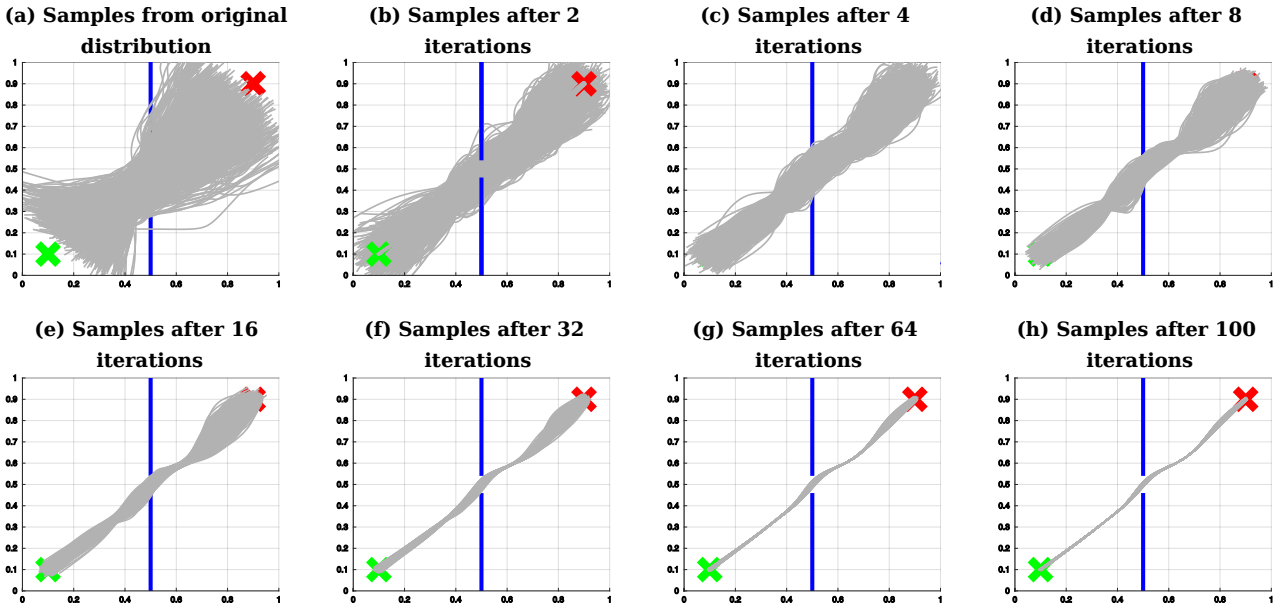


Figure 5.7.: Example of policy search with relevance weighting. The proposed algorithm finds a distribution over trajectories that start and end at the correct positions (represented by the green x and by the red x, respectively) and do not hit the wall (represented by the blue lines).

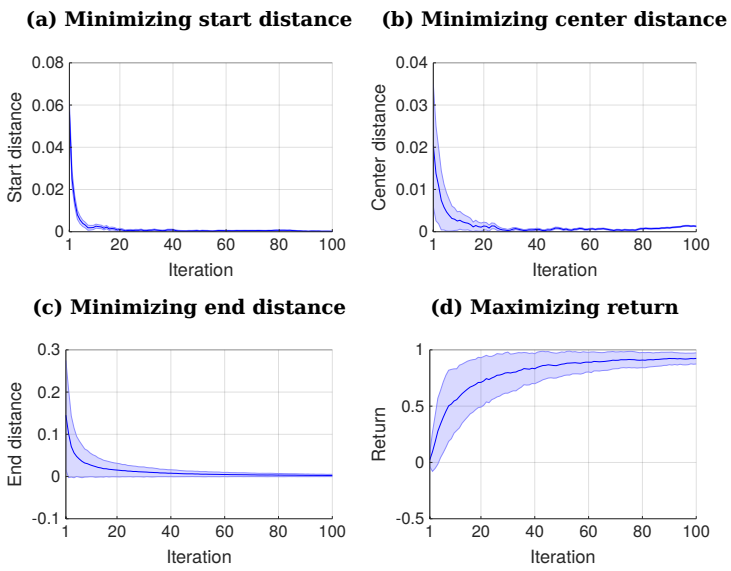


Figure 5.8.: Iteration versus distances and iteration versus returns. The plots represent mean and two times the standard deviation. All the distances to the points of interest decrease to 0 or close to it with the number of iterations. A return function given by $\exp\left(-\beta_{\text{policy}}\left(d_{start}+d_{center}+d_{end}\right)\right)$ increases with the number of iterations. Here, β_{policy} is a parameter which can be determined with line search, d_{start} is the distance to the start, d_{center} is the distance to the center and d_{end} is the distance to the end.

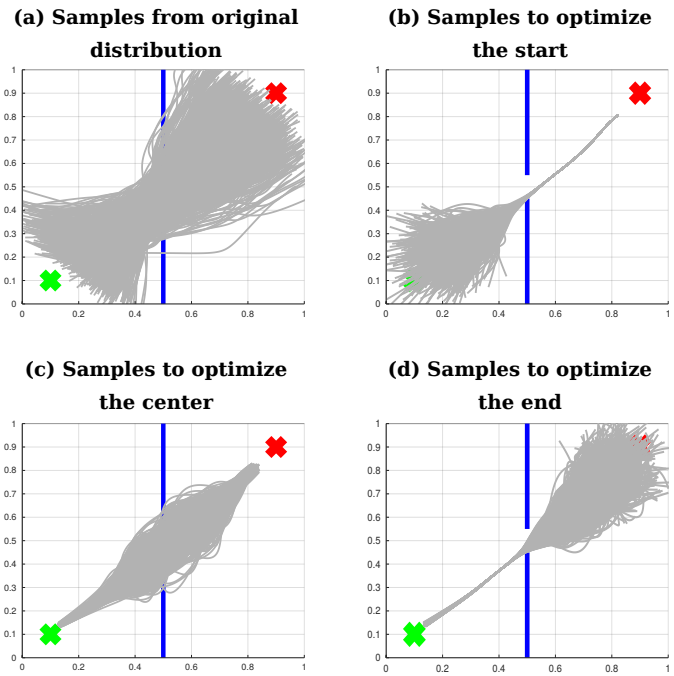


Figure 5.9.: Sampling with relevance weighting. **(a)** Samples from the original distribution. **(b)** Samples to optimize the distribution over trajectories with respect to beginning at the start position. **(c)** Samples to optimize the distribution over trajectories with respect to passing through the center of the window. **(d)** Samples to optimize the distribution over trajectories with respect to reaching the end position. The proposed algorithm explores for each objective a large range of values for the policy parameters that are relevant to that objective, while sampling values close to the mean for the other policy parameters. The variance of the irrelevant parameters is recovered according to Equation 5.43. Therefore, after optimizing for each objective, the distribution over the relevant parameters is updated, while the distribution over the irrelevant parameters is preserved.

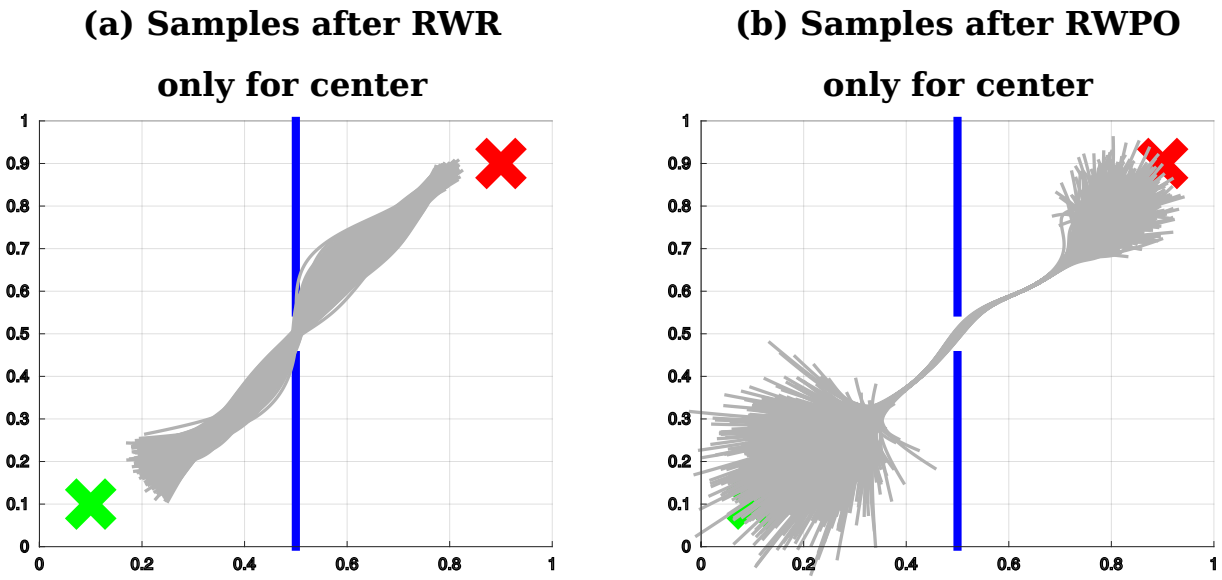


Figure 5.10.: **(a)** Sample trajectories after using Reward-weighted Regression (RWR) to optimize the distribution over trajectories with respect to passing through the center of the window. **(b)** Sample trajectories after using Relevance Weighted Policy Optimization (RWPO) to optimize the distribution over trajectories with respect to the same objective, using the same reward function. In contrast to RWR, RWPO finds a better policy to avoid hitting the wall and does not shrink the variance of parts of the trajectories that are far away from the region of interest.

Figure 5.10(a) shows a distribution over trajectories learned by Reward-weighted Regression (RWR) optimizing only for passing through the center of the window. Figure 5.10(b) shows the solution of Relevance Weighted Policy Optimization (RWPO) for this same optimization problem. RWPO's solution achieves the objective with higher accuracy and preserves a large variance for parts of the trajectory that do not influence the objective.
Finally, Figure 5.11 shows a comparison between Reward-weighted Regression (RWR), sequential Reward-weighted Regression (sRWR) and Relevance Weighted Policy Optimization (RWPO). RWR used here a reward function of the form $R = \exp\left(-\beta_{\text{policy}}\left(d_{start} + d_{center} + d_{end}\right)\right)$, while sRWR and RWPO used one reward function for each objective. The reward functions of sRWR and RWPO were

$$R_{\text{start}} = \exp\left(-\beta_{\text{policy}}\left(d_{start}\right)\right), \tag{5.44}$$

$$R_{\text{center}} = \exp\left(-\beta_{\text{policy}}\left(d_{center}\right)\right), \tag{5.45}$$

$$R_{\text{end}} = \exp\left(-\beta_{\text{policy}}\left(d_{end}\right)\right). \tag{5.46}$$

The distribution of trajectories computed by RWPO can better satisfy the criteria of this problem than the distributions computed by the other two methods.

5.6 Experiments

We demonstrate our method to assist the practice of motor skills by humans with the task of writing Japanese characters. Moreover, an experiment involving a haptic device, the Haption Virtuose 6D, demonstrates how our method can be used to give haptic feedback to the user, guiding him/her towards correct movements according to certain performance criteria even in the absence of expert demonstrations.

5.6.1 Teaching Japanese Characters

In these experiments, first, a human provided with a computer mouse 10 demonstrations of a certain Japanese character composed of multiple strokes. Our system aligned these demonstrations in space and time. Afterward, a human provided a new trajectory. This new trajectory was also aligned in space and time by our system with respect to the

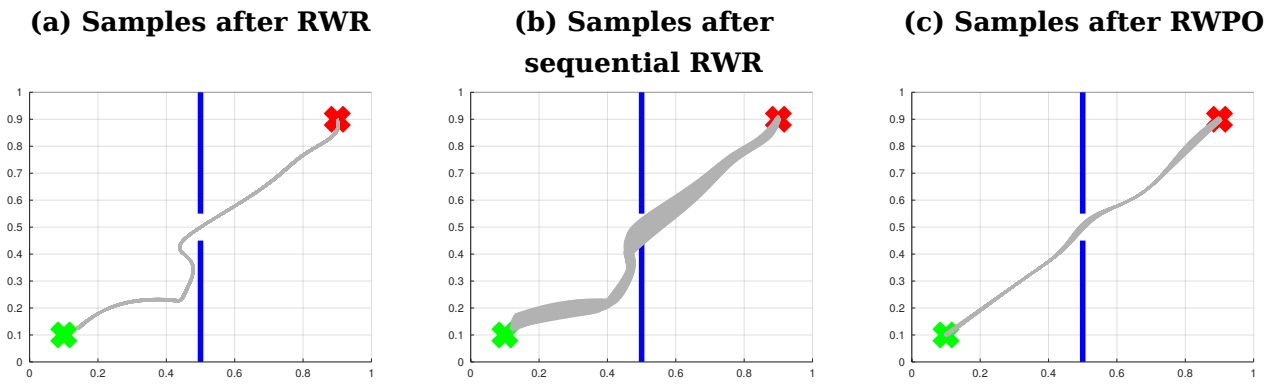


Figure 5.11.: Comparison between Reward-weighted Regression (RWR), sequential Reward-weighted Regression (sRWR) and Relevance Weighted Policy Optimization (RWPO). This time, the three algorithms optimize the distribution over trajectories with respect to all objectives. **(a)** Distribution after optimization with RWR, which uses a single reward function with a term for each objective. **(b)** Distribution after optimization using sRWR, which optimizes for each objective sequentially and has a reward function for each objective. **(c)** Distribution after optimization using RWPO, which uses the concept of relevance functions and optimizes for each objective sequentially with a reward function for each objective.

demonstrations. Once all the demonstrations and the new trajectory had been time-aligned, our system computed a probability distribution over the demonstrations. Based on the probability density function evaluated at each position of the new trajectory in comparison to the probability density function evaluated at corresponding positions along the mean trajectory, our system computed a score. This score was then used to highlight parts of the new trajectory that do not fit the distribution over demonstrations with a high probability.
Figure 5.12 shows some examples of feedback provided by our system. The new trajectory provided by the user is also aligned in space and time. Therefore the absolute position of his/her character and its scale are not relevant. The speed profile of the new trajectory can also be different from the speed profile of the demonstrations. Figure 5.12 shows the new trajectories already after alignment in space and time.

5.6.2 Haptic Feedback

When learning complex movements in a 3D environment or perhaps when manipulating objects, haptic feedback may give the human information about how to adapt his/her movements that would be difficult to extract only from visual feedback. Therefore, we investigated how to give haptic feedback based on a probability distribution over trajectories possibly provided by an instructor or resulting from a reinforcement learning algorithm. This study was carried out in accordance with the recommendations of the Declaration of Helsinki in its latest version. The protocol was approved by the Ethical Committee of the Technische Universität Darmstadt. All participants provided written informed consent before participation.
In this user experiment, users had to use the Haption Virtuose 6D device to teleoperate a small cube in a 3D environment (See Figure 5.14(a)). The users were instructed to begin at the position marked by the yellow sphere, pass through the center of the window in the wall and end at the position marked by the blue sphere. Moreover, it was allowed, at any time, to rotate the virtual environment, zoom in and zoom out using the computer mouse. Five users took part in our experiments: two females and three males, between 27 and 29 years old. Three users had not used the Virtuose 6D before, while two users did have some experience with it.
In the first phase of the experiment, users tried to perform the task ten times without force feedback. Right before each trial, the user pressed a button on the handle of the haptic device, indicating to our system when to start recording the trajectory. By pressing this same button another time, by the end of the trajectory, the user indicated to our system when to finish recording. The users were then instructed to move the cube back to the start position and perform another trial. This procedure would be repeated until ten trajectories had been recorded. Afterward, our system would align them in time and compute a probability distribution over them. Figure 5.14(b) shows a visualization of the distribution over trajectories of one user after this phase. Subsequently, our system optimized this distribution over trajectories using the proposed Relevance Weighted Policy Optimization (RWPO) algorithm. An example of trajectories before and after RWPO is depicted in Figure 5.13. Figure 5.14(c) shows the optimized distribution over trajectories given the initial distribution shown in Figure 5.14(b). After optimizing the distribution over trajectories, our system used it to give force feedback to

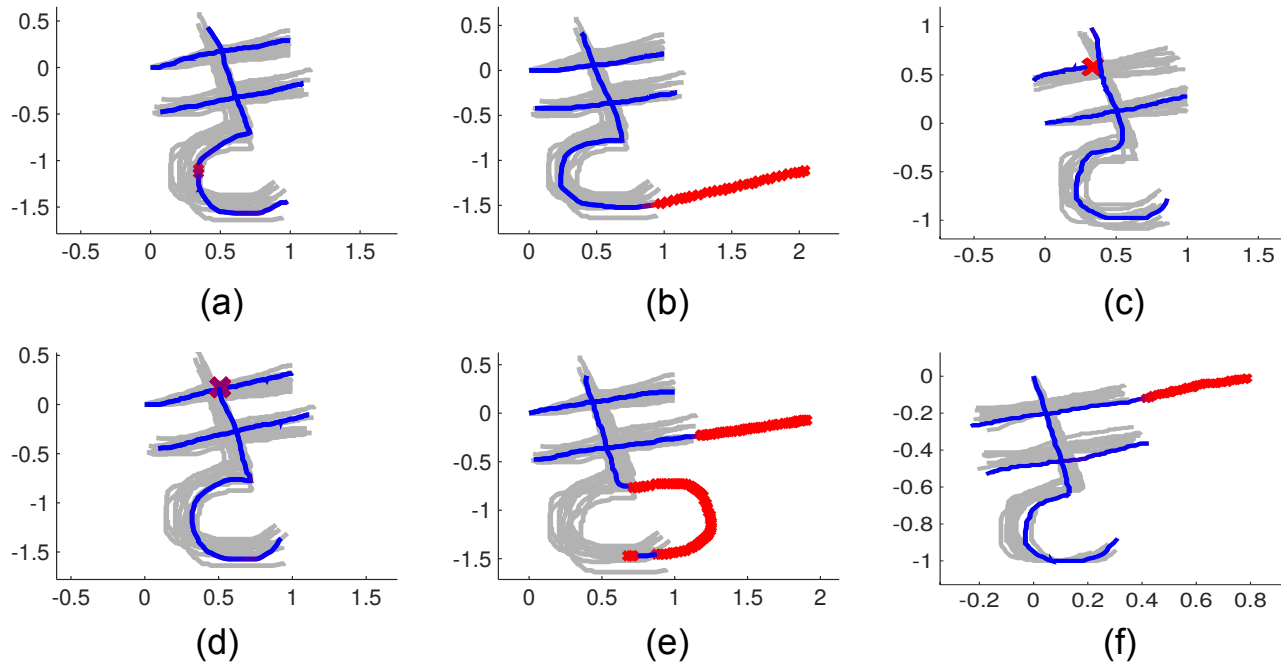


Figure 5.12.: The demonstrations after rescaling, repositioning and time-alignment are depicted in light gray. Parts of a new trajectory that are considered correct are depicted in blue. Parts of a new trajectory that are considered wrong are marked with red x's . **(a)** Instance with a small mistake in the third stroke. **(b)** Third stroke goes further than it should. **(c)** First stroke is too short. **(d)** Third stroke starts too low. **(e)** Second stroke is too long and third stroke has its arch in the wrong direction. **(f)** First stroke is too long.

the user according to the method explained in Section 5.4. The users were requested to try to perform the task with force feedback ten times using the aforementioned procedure to record the trajectories.
Results showing the performance of the users with and without force feedback are presented in Figure 5.15. The use of force feedback did not greatly influence the distance to the start because the force feedback was activated only when the user pressed a button, right before starting to move. The start distance of the third trial of user 2 with force feedback is an outlier. This outlier was due to the user starting far away from the start position. The use of force feedback decreased the distance to the center of the window for all users and the distance to the end for three out of five users. The plots of trial versus distances indicate that the users did not achieve better performance with the force feedback only due to training through repetition because there is a clear difference between the performance in trials with force feedback and the performance in trials without force feedback.
A 2 (feedback) × 3 (distance measures) repeated-measures ANOVA was conducted to test the performance differences between the conditions with and without force feedback. The results reveal significant main effects of feedback ($F(1,4) = 16.31$; $p < 0.05$; $\eta_p^2 = 0.80$) and distance measure ($F(1,5) = 12.93$; $p < 0.05$; $\eta_p^2 = 0.76$; after ϵ correction for lack of sphericity) as well as a significant interaction of feedback × distance measure ($F(1,5) = 10.10$; $p < 0.05$; $\eta_p^2 = 0.72$; after ϵ correction for lack of sphericity). Follow-up one-factor (feedback) repeated-measures ANOVA revealed a significant difference for distance to the center ($F(1,4) = 57.32$; $p < 0.05$; $\eta_p^2 = 0.94$), but not for distance to the start ($F(1,4) = 0.11$; $p = 0.76$) and end ($F(1,4) = 3.61$; $p = 0.13$), respectively. Therefore, feedback had only a significant and strong effect on the distance to the center. However, due to the small sample, the distance to the end test was slightly underpowered ($1 - \beta = 0.798$; corresponding to $\eta_p^2 = 0.474$). Thus, we conclude that force feedback has a differential influence on performance. Whereas force feedback does not influence initial error, later errors are expected to be substantially influenced by force feedback. However, further studies with bigger samples are required to confirm this conclusion.
As can be seen in Figure 5.15(c), users 3 and 5 were able to reach the desired end position with approximately the same accuracy with and without force feedback. Moreover, it has not been enforced in our experiments that users really finish their trials at the end position. Users have been instructed to finish their trials both with and without force feedback whenever they thought they have reached the end position. We could instead, for the trials with force feedback, instruct users to stop only when they feel force feedback contrary to the continuation of the trajectory, which could potentially help to minimize the variance of the end distance with force feedback as observed for users 1 and 2.

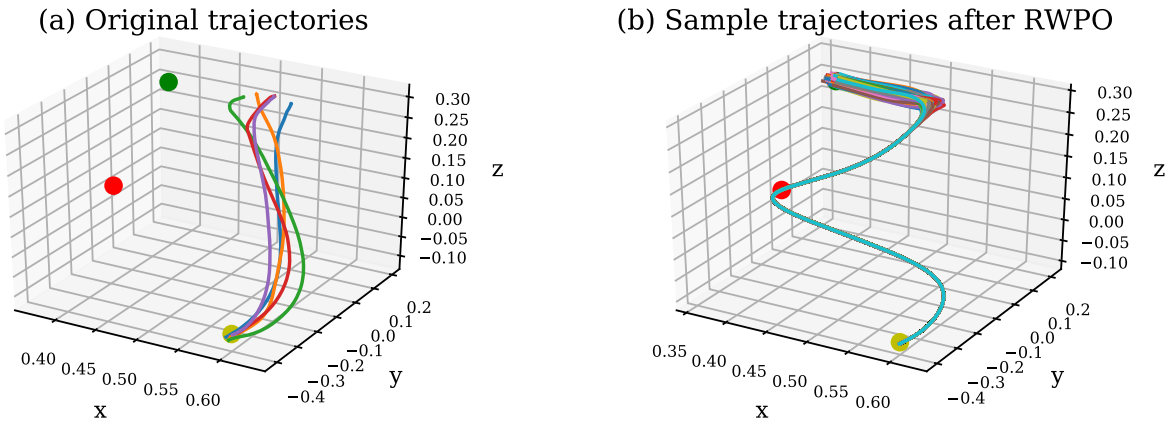


Figure 5.13.: (a) Original trajectories. **(b)** Sample trajectories after Relevance Weighted Policy Optimization (RWPO).

(a) Virtual environment **(b) Before reinforcement learning** **(c) After reinforcement learning**

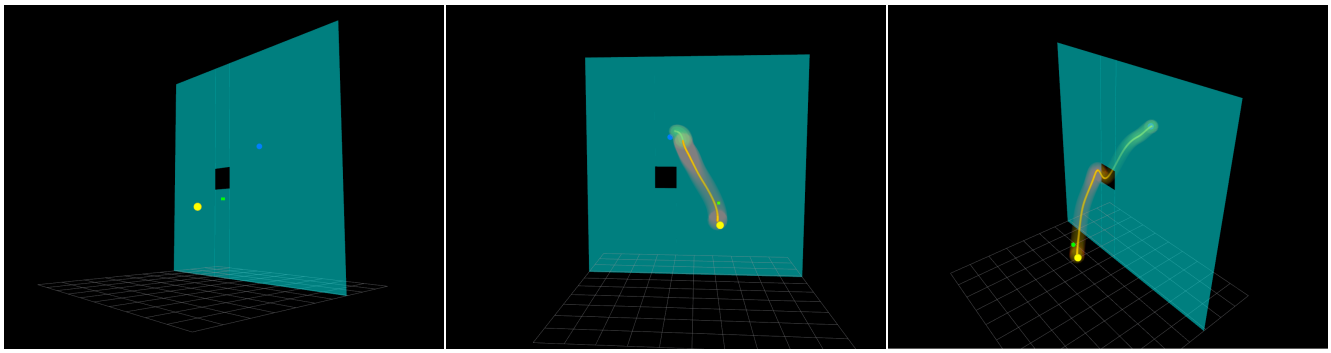


Figure 5.14.: (a) Virtual environment. The goal of the user is to teleoperate the green cube to move it from the position marked by the yellow sphere to the position marked by the blue sphere through the window to not hit the wall. **(b)** Distribution over trajectories before reinforcement learning. **(c)** Distribution over trajectories after reinforcement learning using the proposed Relevance Weighted Policy Optimization (RWPO) algorithm.

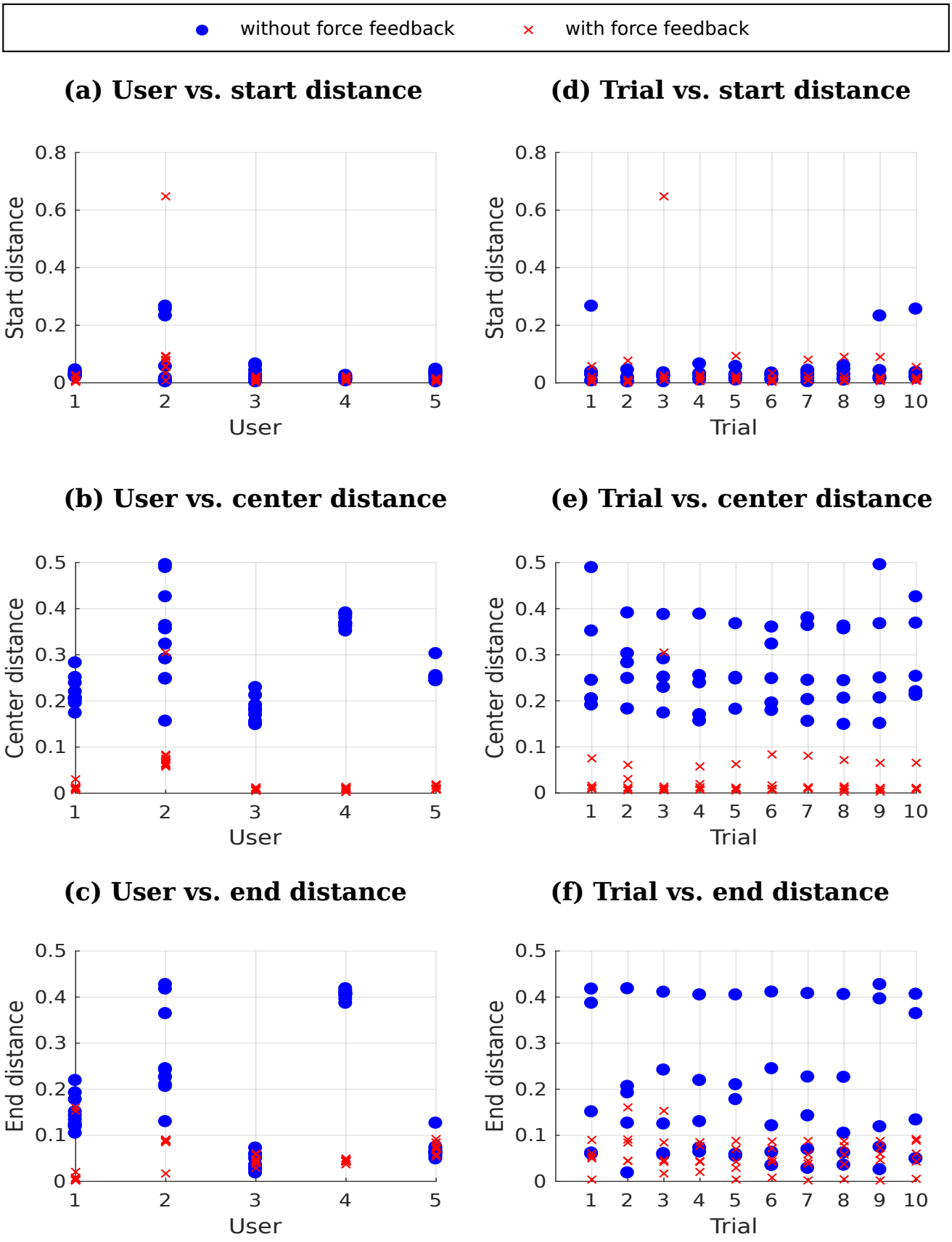


Figure 5.15.: Distances without force feedback and with force feedback. **(a,b,c)** User versus distances, where each data point corresponds to a different trial. **(d,e,f)** Trial versus distances, where each data point corresponds to a different user.

5.7 Epilogue

This chapter presents a probabilistic approach for assisting the practice and the execution of motor tasks by humans. The method here presented addresses the alignment in space and time of trajectories representing different executions of a motor task, possibly composed of multiple strokes. Moreover, it addresses building a probability distribution over demonstrations provided by an expert, which can then be used to assess a new execution of a motor task by a user. When no expert demonstrations are available, our system uses a novel reinforcement learning algorithm to learn suitable distributions over trajectories given performance criteria. This novel algorithm, named Relevance Weighted Policy Optimization, is able to solve optimization problems with multiple objectives by introducing the concept of relevance functions of the policy parameters. The relevance functions determine how the policy parameters are sampled when optimizing the policy for each objective.

We evaluated our framework for providing visual feedback to a user practicing the writing of Japanese characters using a computer mouse. Moreover, we demonstrated how our framework can provide force feedback to a user, guiding him/her towards correct movements in a teleoperation task involving a haptic device and a 3D environment.

Our algorithm to give visual feedback to the user practicing Japanese characters has still some limitations that could possibly be addressed by introducing a few heuristics. For example, the current algorithm assumes that the orientation of the characters is approximately the same. A correct character written in a different orientation would be deemed wrong by our algorithm. Procrustes Analysis [77] provides a solution to align objects with different orientations. Our algorithm could be extended in the future with a similar technique to give meaningful feedback to the user regardless of the orientation of the characters.

In our system, the user has to enter the correct number of strokes to receive feedback. For example, if the user is practicing a character composed of three strokes, the system waits until the user has drawn three strokes. Furthermore, the user has to draw the characters in the right order to get meaningful feedback, otherwise, strokes that do not really correspond to each other are compared. These limitations can potentially be addressed by analyzing multiple possible alignments and multiple possible stroke orders, giving feedback to the user according to the alignment and order that result in the best score.

Our current framework can give the user feedback concerning the shape of a movement, but not concerning its speed. In previous work [83], we have demonstrated how to learn distributions over shape and phase parameters to represent multiple trajectories with multiple speed profiles. Instead of giving the user force feedback towards the closest position along the mean trajectory, the distribution over phase parameters could be used to determine the speed of the attractor along the mean trajectory and how much the user is allowed to deviate from that speed. This extension shall be made in future work.

The framework proposed here could be applied in a scenario where a human would hold a brush connected to a robot arm. The robot could give the user force feedback to help him/her learn both the position and the orientation of the brush when writing calligraphy.

Especially if our framework can be extended to give feedback to the user concerning the right speed of a movement, it could potentially be applied in sports. This work could, for example, help users perform correct movements in weight training, such as in [31, 32]. Another possibility would be to help users train golf swings given expert demonstrations or given optimized probability distributions over swings.

6 Learning Trajectory Distributions for Assisted Teleoperation and Path Planning

Several approaches have been proposed to assist humans in co-manipulation and teleoperation tasks given demonstrated trajectories. However, these approaches are not applicable when the demonstrations are suboptimal or when the generalization capabilities of the learned models cannot cope with the changes in the environment. Nevertheless, in real co-manipulation and teleoperation tasks, the original demonstrations will often be suboptimal and a learning system must be able to cope with new situations. This chapter presents a reinforcement learning algorithm that can be applied to such problems. The proposed algorithm is initialized with a probability distribution of demonstrated trajectories and is based on the concept of relevance functions. We show in this chapter how the relevance of trajectory parameters to optimization objectives is connected with the concept of Pearson correlation. First, we demonstrate the efficacy of our algorithm in static environments by addressing an assisted teleoperation problem. Afterward, we extend this algorithm to deal with dynamic environments by utilizing Gaussian Process regression. The full framework is tested in two online planning problems: one with a point particle and the other with a 7-DoF robot arm.

6.1 Prologue

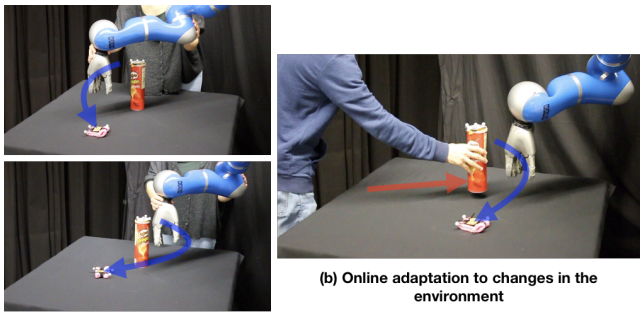
Learning from demonstrations is a promising approach towards human-robot co-manipulation and teleoperation. With this approach, a user can easily demonstrate trajectories to a robot, for instance in gravity compensation mode. Subsequently, the robot fits a model to these trajectories, which allows it to assist the user in the execution of repetitive tasks. The robot assistance can potentially reduce the cognitive load in the user and prevent unintended collisions. Moreover, training a teleoperated robot through demonstrations can give it a certain degree of autonomy, which is desirable in the face of communication latency and intermittency.

Recently, methods have been proposed to assist humans in co-manipulation and teleoperation tasks given demonstrated trajectories [26,28,73]. Our work contributes to this field by providing a new reinforcement learning algorithm, **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO), to improve upon demonstrated trajectories when these are suboptimal or when solutions to new situations must be found. These trajectories need to be optimized with respect to objectives such as minimizing distances to via points, keeping a certain minimum distance from obstacles, achieving minimal length, minimal jerk, etc.

In [84], we have introduced the concept of relevance to optimize trajectory distributions. By using this concept, it is possible to preserve the variance of trajectory parameters that are not relevant to the objective currently being optimized for. This property is helpful for optimizing trajectories sequentially with respect to multiple objectives because the search for optimal parameters for a certain objective does not disturb other parameters that are irrelevant to the objective currently under consideration. In that work, a relevance function was represented by a weighted sum of basis functions. The relevance weights were learned through an iterative process. The new algorithm presented in this chapter, PRO, is based on the insight that the Pearson correlation coefficient can be used to determine how each trajectory parameter influences each objective. It does not require designing basis functions for the relevance. Moreover, the relevance is now determined in one shot in contrast to the previous iterative approach. The efficacy of the proposed algorithm is demonstrated in an assisted teleoperation experiment involving a haptic device, the Haption Virtuose 6D (See Fig. 6.2).

Finally, we extend PRO with Gaussian Processes (GP) regression to cope with dynamic environments. A GP is initialized with demonstrations and prior knowledge. Given a new environment, it outputs a distribution of trajectories which guides the movements of a robot to solve a certain task. PRO is used to optimize upon the GP inferences, gradually improving the mapping from environment to trajectory distribution. After a phase of self-optimization, our learning system is able to compute successful trajectories on the fly face to changes in the environment. This chapter presents applications of this framework in two problems. The first problem involves a point particle that needs to achieve a dynamic target while avoiding moving walls. The second one consists in controlling a 7-DoF robot arm to reach a target while avoiding a cylinder on a table. Both the position of the target and the position of the cylinder can be changed by a human (See Fig. 6.1).

The main contribution of this chapter is a new algorithm that learns trajectory distributions to solve planning tasks both in static and dynamic environments. The learned trajectory distributions are appealing because they function as virtual guides for users in shared-autonomy tasks, e.g. assisted teleoperation, as it is demonstrated in our experiments.



(a) Demonstrations

(b) Online adaptation to changes in the environment

Figure 6.1.: In this experiment, the robot has to reach the target (pink object) while avoiding the obstacle (Pringles can). **(a)** Demonstrations. **(b)** Our learning system is able to adapt on the fly a distribution of trajectories to changes in the environment. This distribution can be used for path planning tasks as well as for co-manipulation and assisted teleoperation in dynamic environments.

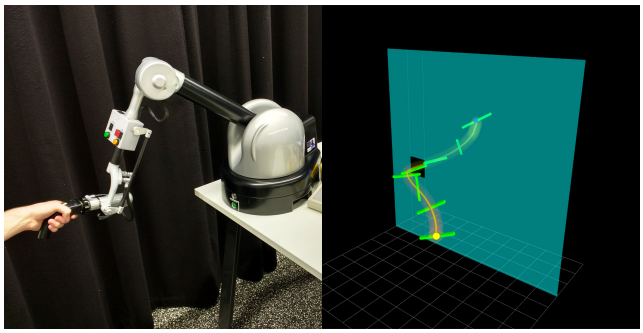


Figure 6.2.: A haptic device, the Haption Virtuose 6D, is operated by a user to move a beam in a virtual environment. The haptic device uses a trajectory distribution learned with **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO) to assist the user in moving the beam from a start position and orientation to an end position and orientation through a window in the wall.

6.2 Related Work

To assist users in teleoperation or potentially in co-manipulation tasks given suboptimal demonstrations, our algorithm has to be able to refine continuous paths to avoid obstacles, pass through via points, etc. CHOMP [85] and STOMP [86] are two prominent methods for continuous path refinement. CHOMP is a gradient-based optimization technique to minimize a cost function while changing an initial trajectory as little as possible. STOMP is a gradient-free, stochastic trajectory optimization technique based on Policy Improvement with Path Integrals (PI^2) [87]. Our algorithm, PRO, presents some similarities to STOMP. It is a gradient-free, stochastic trajectory optimization technique based on Reward Weighted Regression (RWR) [71]. While in STOMP, trajectories are generated by perturbing an initial trajectory with a certain noise, in our work, a distribution of trajectories based on demonstrations (and potentially also on prior knowledge) is optimized.

An effective method for assisting users in shared-autonomy tasks, e.g. co-manipulation, with probabilistic models learned from demonstrations has been proposed in [73]. In that paper, Gaussian mixture models [62] are used to create multiple probabilistic virtual guides, which constrain the movements of the user to a region close to the demonstrations.

As in our work, in [26], probabilistic models are used to assist users in teleoperation tasks with shared control. In that work, task-parameterized Gaussian mixture models (TP-GMMs) [57] have been used to encode the probability distribution of demonstrated trajectories. Gaussian mixture regression (GMR) [57] has been used to generate a behavior according to the learned model. The learning agent assists the user with the teleoperation of a device to scan a surface.

Our work is in line with [73] and [26], with the important difference that our approach addresses cases where demonstrations are suboptimal or when the learned model cannot generalize well enough to a new scenario. This is possible due to the optimization of the original distribution through reinforcement learning. An approach for improving upon suboptimal initial demonstrations is presented in [27]. Nevertheless, that approach is based on iterative refinement by the human user instead of reinforcement learning.

Learning from demonstrations has also been applied in supervisory control. In this paradigm, after training, the remote system can execute a task autonomously, needing only high-level task goals to be provided by the user. In [28], task-parameterized hidden semi-Markov models (TP-HSMMs) are used to build probabilistic models of manipulation motions and Model Predictive Control (MPC) is used to execute these motions. In that work, TP-HSMMs have been shown to generalize better to changes in the environment than Probabilistic Movement Primitives (ProMPs) [88], which are used in our work. We believe that our work can contribute to enhancing the generalization capabilities of frameworks using probabilistic models such as TP-HSMM and ProMP by using reinforcement learning to let the remote system look for solutions to new tasks by trial and error.

Previous works have proposed approaches to adapt initial movements learned from demonstrations to new environments with obstacles [89–91]. As a contribution to this area of research, our approach creates in an offline self-optimization procedure a mapping between environment configurations and trajectory distributions, which can be used to online adapt these distributions in dynamic environments. This is achieved by using Gaussian Process (GP) regression to map variables describing the environment to parameters (mean vector and covariance matrix) of a probability distribution of

trajectories in the form of a ProMP. Our proposed reinforcement learning algorithm, **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO), is used to iteratively refine this mapping. In a sense, PRO is providing the GPs with new optimized ProMPs for any given environment, which gradually improves the mapping. PRO performs Reinforcement Learning while GP regression performs Supervised Learning. This process resembles the way Guided Policy Search (GPS) [92] uses trajectory optimization in combination with the constraint that the actions output by a Convolutional Neural Network (CNN) must track the optimized trajectories. In our approach, PRO assumes the role of the trajectory optimizer while the GPs assumes the role of the CNN. In contrast to GPS, while the CNN outputs actions for any given state, in our approach, GP regression outputs ProMPs (distributions of trajectories) for any given environment. These distributions are used in our work as virtual guides that can constrain the movements of the human and the robot to certain regions of the state space.

6.3 Pearson-Correlation-Based Relevance Weighted Policy Optimization (PRO)

This section explains PRO, a policy search algorithm which uses the relevance of each trajectory parameter to each optimization objective to optimize policies in the form of trajectory distributions. The relevance is computed in one shot by using Pearson correlation coefficients.

6.3.1 Relevance Functions

PRO is a stochastic policy search algorithm based on Reward Weighted Regression (RWR) [71]. In each iteration of RWR applied to trajectory optimization, trajectory parameters are sampled from a probability distribution, e.g. a Gaussian. Subsequently, the parameters of that probability distribution, e.g. mean vector and covariance matrix, are optimized to maximize the expected reward.

In PRO, the relevance of each trajectory parameter to each optimization objective is estimated. This information is then used to determine how the trajectory parameters should be sampled when optimizing the distribution of trajectory parameters with respect to each objective. This procedure prevents undesirable changes in the distribution of trajectory parameters that do not influence the objective under consideration. On the other hand, only the trajectory parameters that actually influence the objective under consideration are subject to exploration in the sampling procedure. Fig. 6.3 illustrates the difference between RWR and PRO.

The key observation in PRO is that the relevance of the trajectory parameter w_n to the objective o, denoted by $f_o(n)$, can be represented by the absolute value $|\rho_{n,o}|$ of the Pearson correlation coefficient

$$\rho_{n,o} = \frac{\mathrm{cov}(w_n, o)}{\sigma_{w_n}\sigma_o}, \tag{6.1}$$

where $\mathrm{cov}(w_n, o)$ is the covariance between w_n and the value of the objective o, σ_{w_n} is the standard deviation of w_n and σ_o is the standard deviation of the values of the objective o. The values $\rho_{n,o}$ can be computed from samples of $\boldsymbol{w} = [w_1, \cdots, w_N]^\top$, where N is the number of trajectory parameters. In this work, the samples $\boldsymbol{w}$ to compute $\rho_{n,o}$ are drawn from a Gaussian with mean vector $\boldsymbol{\mu_w}$ and covariance matrix $\sigma^2_{\text{relevance}}\boldsymbol{I_{N\times N}}$. The mean $\boldsymbol{\mu_w}$ can be based on demonstrations and $\sigma^2_{\text{relevance}}$ is chosen by the user to produce small disturbances around the mean. The underlying assumption in this method of quantifying relevance is that the trajectory parameters are locally linearly correlated with the optimization objectives. This is why $\sigma^2_{\text{relevance}}$ needs to be small. For each $\boldsymbol{w}$, the value of each objective o is computed. Given the samples $\boldsymbol{w}$ and the corresponding objective values o, the computation of $\rho_{n,o}$ is straightforward and can be implemented with a single line of code using libraries such as NumPy.

The Pearson correlation coefficient $\rho_{X,Y}$ of any two random variables X and Y is a measure of the linear correlation between X and Y and $-1 \leq \rho_{X,Y} \leq 1$. Thus the relevance function $f = |\rho_{X,Y}|$ is such that $0 \leq f \leq 1$. The relevance $f_o(n)$ of w_n to the objective o expresses how strongly changes in w_n are linearly correlated to changes in the values of the objective o. In practice, to explore a large range of parameter values, it is helpful to normalize the relevance functions $f_o(n)$ such that their maximum value is 1 instead of a smaller value. The normalized relevance functions are thus given by $\frac{f_o(n)}{\max_n f_o(n)}$. Assuming our probability distribution of trajectory parameters w_n is a Gaussian, i.e. $\boldsymbol{w} \sim \mathcal{N}\left(\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}\right)$, PRO samples w_n from the distribution $\mathcal{N}\left(\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}^{f_o}\right)$, where

$$\boldsymbol{\Sigma_w}^{f_o} = \mathrm{diag}\left(\sigma^2_{w_1} f_o(1), \cdots, \sigma^2_{w_N} f_o(N)\right). \tag{6.2}$$

In our work, the initial distribution $\mathcal{N}\left(\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}\right)$ is based on demonstrations.

6.3.2 Optimization of Trajectory Distributions using Relevance Functions

Once a number S of trajectory parameter vectors $\boldsymbol{w}$ have been sampled from $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}^{f_o}\right)$, Reward Weighted Regression (RWR) is used to optimize the mean $\boldsymbol{\mu}_{\boldsymbol{w}}$ and the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{w}}^{f_o}$ of this distribution to maximize the expected reward. The optimization problem

$$\{\boldsymbol{\mu}_{\boldsymbol{w}}^{k+1}, \boldsymbol{C}^{k+1}\} = \underset{\{\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}{}^{f_o}\}}{\arg\max} \sum_{i=1}^{S} R_{o,i} \mathcal{N}\left(\boldsymbol{w}_i; \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}{}^{f_o}\right) \tag{6.3}$$

has the solution

$$\boldsymbol{\mu}_{\boldsymbol{w}}^{k+1} = \frac{\sum_{i=1}^{S} R_{o,i} \boldsymbol{w}_i}{\sum_{i=1}^{S} R_{o,i}}, \qquad \boldsymbol{C}^{k+1} = \frac{\sum_{i=1}^{S} R_{o,i} \left(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}}^{k}\right)\left(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}}^{k}\right)^{\top}}{\sum_{i=1}^{S} R_{o,i}}. \tag{6.4}$$

The variable k in the expressions above represents the iterations of the algorithm. The variable $R_{o,i}$ represents the reward with respect to objective o obtained by the sampled trajectory i. The reward is non-negative and usually has the form $R_{o,i} = \exp(-\beta o(i))$, where $o(i)$ is the value obtained by the sampled trajectory i for objective o and β is a hyperparameter chosen by the user.
Finally, the new covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ is determined. It is a diagonal matrix with the variances in the diagonal given by

$$\sigma_{w_n,k+1}^2 = (1 - f_o(n))\,\sigma_{w_n,k}^2 + f_o(n)\,\boldsymbol{C}_{nn}^{k+1}, \tag{6.5}$$

where $\sigma_{w_n,k}^2$ is the variance of w_n in iteration k and $\boldsymbol{C}_{nn}^{k+1}$ is the element at row n and column n of the covariance matrix $\boldsymbol{C}^{k+1}$.
Equation (6.5) keeps the variance of irrelevant trajectory parameters unchanged and updates the variance of relevant trajectory parameters. If $f_o(n) = 0$, for example, $\sigma_{w_n,k+1}^2 = \sigma_{w_n,k}^2$, i.e. the variance of w_n at iteration $k+1$ is the same as at iteration k. On the other hand, if $f_o(n) = 1$, $\sigma_{w_n,k+1}^2 = \boldsymbol{C}_{nn}^{k+1}$, i.e. the variance of w_n at iteration $k+1$ is the result of the RWR optimization at iteration k, yielding $\boldsymbol{C}_{nn}^{k+1}$. For other relevance values, which must lie by definition between 0 and 1, the new variance is a weighted average of its previous value and the optimized one.
Algorithms 7 and 8 present a description of PRO in the form of pseudocode. Fig. 6.4 shows a comparison between the algorithm proposed in [84], Relevance Weighted Policy Optimization (RWPO), and the one proposed in this chapter, PRO. PRO has two main advantages over RWPO: 1) in PRO, it is not necessary to design problem-specific basis functions for the relevance; 2) The relevance computation in PRO is performed in one shot, which is much faster than the iterative procedure used in RWPO. Fast computation of the relevance is crucial because, in general, the relevance may need to be reevaluated during the optimization of the trajectory distributions. This necessity is due to the fact that the relevance is computed from samples of a Gaussian with mean $\boldsymbol{\mu}_{\boldsymbol{w}}$ and covariance matrix $\sigma_{\text{relevance}}^2 \boldsymbol{I}_{N\times N}$. Therefore, the relevance depends on $\boldsymbol{\mu}_{\boldsymbol{w}}$, which needs to be optimized.

Algorithm 7 Relevance Learning

Inputs: mean $\boldsymbol{\mu}_{\boldsymbol{w}}$, variance $\sigma_{\text{relevance}}^2$ and objective function corresponding to objective o
Sample trajectory parameters $\boldsymbol{w}$ from $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{w}}, \sigma_{\text{relevance}}^2 \boldsymbol{I}_{N\times N}\right)$
for each sample $\boldsymbol{w}$ **do**
    Compute the value for the objective o obtained by the trajectory with parameters $\boldsymbol{w}$
end for
for each w_n **do**
    Compute the Pearson correlation coefficient $\rho_{n,o}$ (Equation6.1)
    Compute the relevance function $f_o(n) = |\rho_{n,o}|$
    Normalize the relevance function
end for
return the normalized relevance functions $f_o(n)$

Algorithm 8 Pearson-Correlation-Based Relevance Weighted Policy Optimization

Inputs: mean $\boldsymbol{\mu_w}$, covariance $\boldsymbol{\Sigma_w}$ of trajectory parameters $\boldsymbol{w}$,variance $\sigma^2_{\text{relevance}}$ and objective functions
repeat
  for each objective o **do**
    Compute relevance functions $f_o(n)$ (Algorithm 1)
    Compute matrix $\boldsymbol{\Sigma_w}^{f_o}$ (Equation 6.2)
    Sample trajectory parameters $\boldsymbol{w}$ from $\mathcal{N}\left(\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}^{f_o}\right)$
    for each sample $\boldsymbol{w}_i$ **do**
      Compute the reward $R_{o,i}$ of the trajectory with parameters $\boldsymbol{w}_i$ associated with objective o
    end for
    Update $\boldsymbol{\mu_w}$ and compute $\boldsymbol{C}$ (Equations 6.4)
    Update the variances of the trajectory parameters $\sigma^2_{w_n}$ (Equation 6.5)
  end for
until convergence of the rewards $R_{o,i}$
return the mean $\boldsymbol{\mu_w}$ and the variances $\sigma^2_{w_n}$

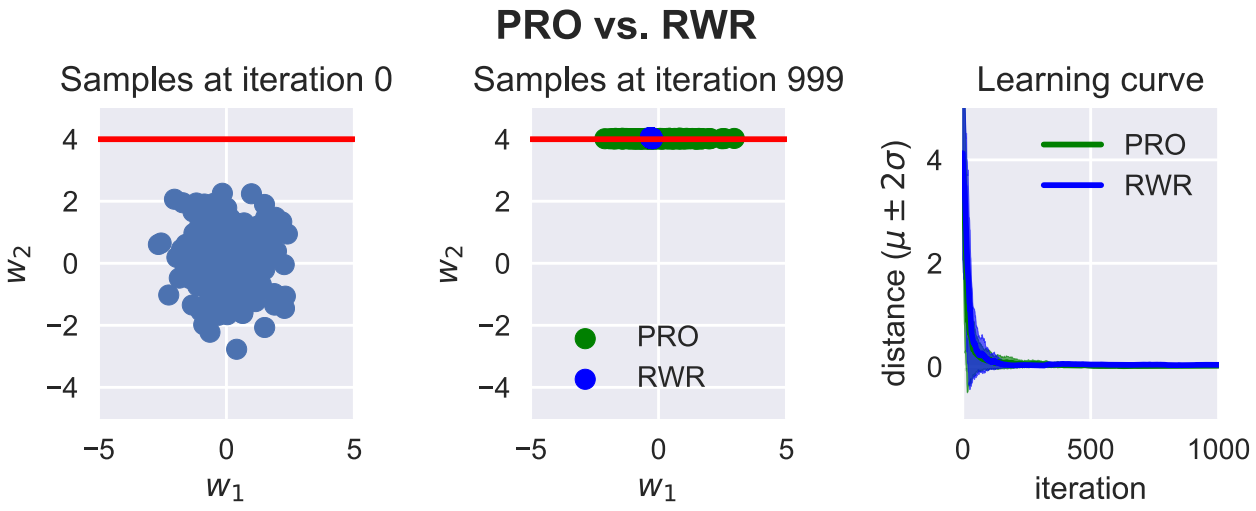


Figure 6.3.: **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO) versus Reward Weighted Regression (RWR). Here, w_1 and w_2 are the trajectory parameters. These trajectory parameters could be for example the weights for some basis functions. For visualization purposes, it is assumed in this example that only two parameters suffice to parameterize the trajectories. The red line represents the region in the space of trajectory parameters where the reward is the maximum. The reward for any point in this space is $R = \exp(-\beta d)$, where β is a hyperparameter chosen by the user and d is the distance between the point and the red line. Both RWR and PRO were applied to optimize a Gaussian distribution of $\boldsymbol{w} = [w_1, w_2]^\top$ with 1000 iterations and 200 samples per iteration. The variances of RWR collapse while PRO is able to keep the variance of w_1 because this parameter is not relevant to this optimization problem. PRO can thus optimize the mean and variance of a certain parameter without disturbing the mean and variance of parameters which are irrelevant to the objective being optimized for. This property is helpful when sequentially optimizing trajectory distributions with respect to several objectives or to conserve as much as possible of the variance of the initial distribution.

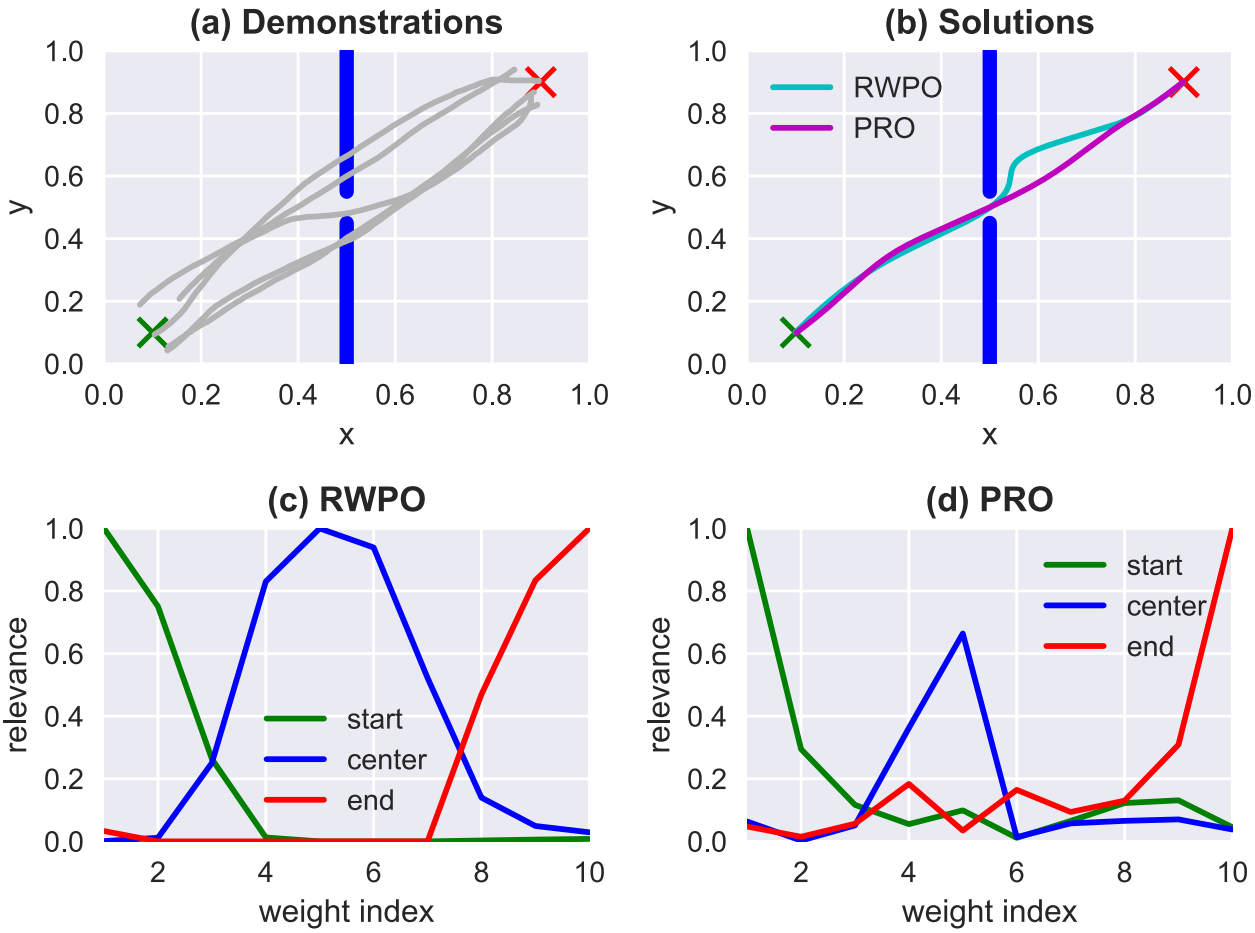


Figure 6.4.: Comparison between Relevance Weighted Policy Optimization (RWPO), proposed in [84], and **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO), proposed in this chapter. In this problem, a trajectory must be found that starts at the green ×, goes through the window in the center and ends at the red ×. **RWPO took 84.47s to learn the relevance functions while PRO took only 0.02s**. Both algorithms were implemented in Python. The machine used for both computations was the same. The hyperparameters for RWPO were the same as used in [84]. PRO used 200 trajectory samples to compute the relevance function. The vast difference in execution time is due to the iterative nature of the relevance computation in RWPO while PRO has a one-shot approach to compute the relevance. **(a)** 5 demonstrations provided by a user. **(b)** Both algorithms find solutions that satisfy the given criteria. The larger deviation from the direct path observed in the solution by RWPO does not represent an error because there is no cost for larger deviations from the direct path. The cost function used by both algorithms is the same and depends only on the distances to the start, window center and end. **(c, d)** Relevance for the 10 weights that parameterize the x trajectory computed by RWPO and by PRO, respectively. The relevance for the 10 weights that parameterize the y trajectory has not been plotted here. Differently from RWPO, PRO does not use basis functions for the relevance.

This work uses Probabilistic Movement Primitives (ProMPs) [88] to represent trajectory distributions. In ProMPs, each trajectory is approximated by a weighted sum of Gaussian basis functions evenly spaced along the time axis. Each trajectory can thus be represented by a vector of weights $\boldsymbol{w} = [w_1, \cdots, w_N]^\top$, where N is the number of Gaussian basis functions. Given a number of demonstrated trajectories for a certain task, a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w})$ of $\boldsymbol{w}$ is computed through Maximum Likelihood Estimation (MLE).

ProMPs allow for computing the posterior probability distribution of trajectories given via points. This operation, however, produces sensible results only if the via points are close to the original ProMP. For our purposes, we need to adapt ProMPs to environment configuration variables like via points and others also when they are very different from the configurations observed during the demonstration phase.

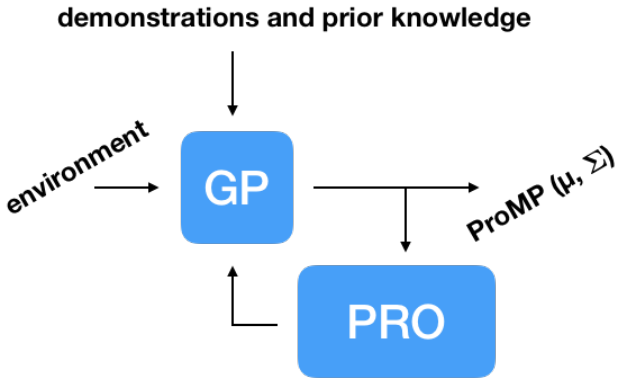


Figure 6.5.: Architecture to adapt trajectory distributions, in this case, ProMPs, to changes in the environment. Our learning system is initialized with demonstrations and potentially with trajectories based on prior knowledge about the task at hand. Gaussian Process (GP) regression is used to infer ProMPs given variables describing the current state of the environment. **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO) optimizes the inferences made by GP regression, updating the dataset of environment configurations and corresponding ProMPs, gradually improving the quality of the inferences.

To adapt ProMPs on the fly to changes in the environment, our learning system must be able to compute these ProMPs quickly. To deal with this challenge, we propose using Gaussian Process (GP) regression to map variables describing the environment to mean vector $\boldsymbol{\mu_w}$ and covariance matrix $\boldsymbol{\Sigma_w}$ of a ProMP. Our learning system is trained according to the following steps: 1) Initialization with demonstrations and prior knowledge; 2) Given a random state of the environment, infer a ProMP using GP regression; 3) PRO optimizes upon the inferred ProMP; 4) Update dataset of environment states and corresponding ProMPs with the solution provided by PRO. Steps 2 to 4 are repeated several times until the learning system is able to solve a given task for a range of possible configurations of the environment. For a task in which a robot needs to move from a start position to an end position while avoiding an obstacle, for example, a suitable initialization based on prior knowledge can be a distribution of trajectories with mean going directly from the start position to the end position and a certain amount of noise for exploration by PRO. Fig. 6.5 depicts our proposed architecture.

The vector of variables describing the current state of the environment is denoted by $\boldsymbol{e}$. The elements of this vector can be for example obstacle positions, via points, target positions, etc.

The user is asked to initialize our learning system by providing multiple demonstrations for each environment configuration $\boldsymbol{e}_m$ in the set $\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_M\}$ containing M different configurations. Based on these demonstrations, the variables $\mu_{w_{n,m}}$ and $\sigma^2_{w_{n,m}}$ are computed through Maximum Likelihood Estimation (MLE), where $n \in \mathbb{N}, m \in \mathbb{N}, 1 \leq n \leq N, 1 \leq m \leq M$. The variables $\mu_{w_{n,m}}$ and $\sigma^2_{w_{n,m}}$ are the mean and the variance, respectively, of weight w_n based on the demonstrations for environment configuration $\boldsymbol{e}_m$. The set of demonstrations can be augmented for additional environment configurations by trajectories based on prior knowledge, as previously mentioned. In this case, the trajectories based on prior knowledge are treated just as the demonstrations directly provided by the user.

Given $\mu_{w_{n,m}}$ and $\sigma^2_{w_{n,m}}, \forall m, 1 \leq m \leq M$, the variables μ_{w_n} and $\sigma^2_{w_n}$ are computed. These variables represent the average mean $\mu_{w_n} = \frac{1}{M}\sum_{m=1}^{M} \mu_{w_{n,m}}$ and the average variance $\sigma^2_{w_n} = \frac{1}{M}\sum_{m=1}^{M} \sigma^2_{w_{n,m}}$ for each weight w_n.

A new environment $\boldsymbol{e}_{\text{new}}$ is sampled at random from a set containing both the environments $\boldsymbol{e}_1, \cdots, \boldsymbol{e}_M$ for which there were demonstrations as well as environments for which there were no demonstrations. Gaussian Process (GP) regression is used to infer the trajectory parameters $w_{n,\text{new}}$ for the new configuration $\boldsymbol{e}_{\text{new}}$, given the demonstrations. There is one Gaussian Process (GP) for each parameter w_n. The GPs use the squared exponential kernel

$k\left(\boldsymbol{e}_i, \boldsymbol{e}_j\right) = \exp\left(-\alpha\left(\boldsymbol{e}_i - \boldsymbol{e}_j\right)^\top\left(\boldsymbol{e}_i - \boldsymbol{e}_j\right)\right), \alpha \in \mathbb{R}, \alpha > 0$. The variables $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ represent any two arbitrary environment configurations. The posterior $p\left(w_{n,\text{new}}|w_{n,1:M}\right) = \mathcal{N}\left(\mu_{w_{n,\text{new}}}, \sigma^2_{w_{n,\text{new}}}\right)$ is a Gaussian with

$$\mu_{w_{n,\text{new}}} = \mu_{w_n} + \boldsymbol{K}_{\text{new},1:M}\left(\boldsymbol{K}_{1:M,1:M} + \boldsymbol{\Sigma}_{w_n}\right)^{-1}\left(\boldsymbol{\mu}_{w_{n,1:M}} - \boldsymbol{\mu}_{w_n}\right), \tag{6.6}$$

$$\sigma^2_{w_{n,\text{new}}} = \boldsymbol{K}_{\text{new,new}} + \sigma^2_{w_n} - \boldsymbol{K}_{\text{new},1:M}\left(\boldsymbol{K}_{1:M,1:M} + \boldsymbol{\Sigma}_{w_n}\right)^{-1}\boldsymbol{K}_{1:M,\text{new}}, \tag{6.7}$$

where $\boldsymbol{\mu}_{w_n} = \left[\mu_{w_n}, \cdots, \mu_{w_n}\right]^\top$ is a column vector with μ_{w_n} repeated M times, $\boldsymbol{\mu}_{w_{n,1:M}} = \left[\mu_{w_{n,1}}, \cdots, \mu_{w_{n,M}}\right]^\top$, the covariance matrix of each GP is

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_{\text{new,new}} & \boldsymbol{K}_{\text{new},1:M} \\ \boldsymbol{K}_{1:M,\text{new}} & \boldsymbol{K}_{1:M,1:M} \end{pmatrix} \tag{6.8}$$

and

$$\boldsymbol{\Sigma}_{w_n} = \text{diag}\left(\sigma^2_{w_{n,1}}, \cdots, \sigma^2_{w_{n,M}}\right). \tag{6.9}$$

The covariance matrix $\boldsymbol{K}$ comprises four blocks. $\boldsymbol{K}_{\text{new,new}}$ is here just the scalar $k(\boldsymbol{e}_{\text{new}}, \boldsymbol{e}_{\text{new}})$. $\boldsymbol{K}_{\text{new},1:M}$ is a row vector with elements $k\left(\boldsymbol{e}_{\text{new}}, \boldsymbol{e}_j\right), j \in \mathbb{N}, 1 \leq j \leq M$. $\boldsymbol{K}_{1:M,\text{new}}$ is a column vector with elements $k(\boldsymbol{e}_i, \boldsymbol{e}_{\text{new}}), i \in \mathbb{N}, 1 \leq i \leq M$. Finally, $\boldsymbol{K}_{1:M,1:M}$ is a matrix with elements $k(i, j)$.
After computation of the posterior distribution given by $\mu_{w_{n,\text{new}}}$ (Eq. 6.6) and $\sigma^2_{w_{n,\text{new}}}$ (Eq. 6.7), PRO optimizes upon this distribution as described in Section 6.3. The dataset of known environments and corresponding trajectory distributions is updated with $\boldsymbol{e}_{\text{new}}$, $\mu_{w_{n,\text{new}}}$ and $\sigma^2_{w_{n,\text{new}}}$, $\forall n, 1 \leq n \leq N$. Subsequently, this entire process is repeated for another $\boldsymbol{e}_{\text{new}}$. After several iterations, as will be shown in the experimental section, the learning system is able to generate successful distributions of trajectories for a pre-defined range of environment configurations.

6.5 Experiments

Three experiments demonstrate the efficacy of our proposed framework. The first experiment demonstrates that PRO can be applied to optimize upon initial failed attempts to solve a teleoperation task. The last two experiments demonstrate how PRO in combination with Gaussian Process (GP) regression can tackle motion planning problems in dynamic environments.

6.5.1 Assisting Humans in a Teleoperation Task

In this experiment, the user manipulates the Haption Virtuose 6D to move a beam in a virtual environment (See Fig. 6.2). This experiment can be seen as a teleoperation task, where the haptic device is the master and the beam is the slave. The goal of the user is to move the beam from a start position and orientation to an end position and orientation through the window without hitting the wall. This task is hard for humans in part due to the difficulty in visually estimating the 3D position and the orientation of the beam.
First, the user tries ten times to perform the task without force feedback. A distribution of trajectories based on the trials of the user is created using a ProMP. Subsequently, PRO is used to optimize this ProMP such that sample trajectories from the optimized ProMP pass through four via points with the right beam orientations to avoid collisions with the wall.
In this experiment, the original optimization problem has been separated into two optimization problems: one taking into consideration only the Cartesian coordinates of the via points and another taking into consideration only the orientation of the beam at each via point. This separation helped PRO to find successful trajectories in this problem. The reward function for both problems is $R_o = \exp\left(-\beta\left(o + \beta_l l + \beta_j j\right)\right)$, with $\beta = 200$, $\beta_l = 0.1$ and $\beta_j = 10^5$. The value for β was empirically determined by trying a few values between 1 and 300. The values for β_l and β_j were determined by trying different powers of 10. The variable o represents the distances to each of the four via points. The two via points closest to the window are computed given the current position of the window. The variables l and j represent the length and the average jerk magnitude of the trajectory, respectively, and are computed by using finite differences. The terms β_l and β_j are used to regulate the importance of the length and average jerk magnitude to the reward. PRO optimized the ProMP in 150 iterations with 200 trajectory samples per iteration.
Fig. 6.6(a) shows the initial trials of a user to solve the task for a given scenario without the assistance of the haptic device. Fig. 6.6(b-f) represents the optimized ProMP, which is used by the haptic device to guide the user with force feedback inverse proportional to the standard deviation.

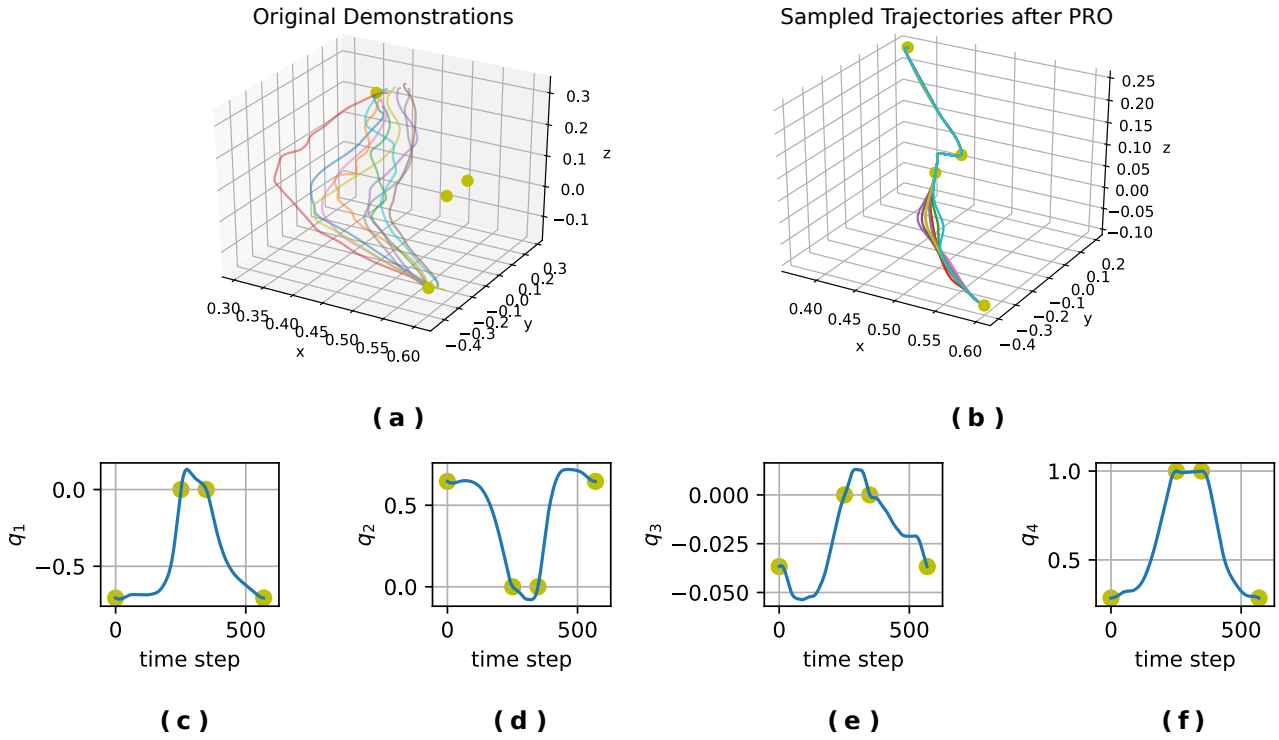


Figure 6.6.: **(a)** Cartesian coordinates of demonstrated trajectories. **(b)** Cartesian coordinates of trajectories sampled from the ProMP optimized with PRO. **(c - f)** Orientations in quaternions of the mean trajectory of the ProMP optimized with PRO. The yellow dots represent via points.

6.5.2 Adaptation in Dynamic Environments — Point Particle

The problem addressed in this section is depicted in Fig. 6.7. First, a human was presented with 30 random environments. The environments differed in $\boldsymbol{c} = (x_c, y_c)$, the position of the center of the hole in the wall, and in $\boldsymbol{g} = (x_g, y_g)$, the position of the end goal (red × in Fig. 6.7). By using a computer mouse, the human provided three demonstrations for each environment.

The random environments $\boldsymbol{e} = \left[x_c, y_c, x_g, y_g\right]^\top$ were uniformly distributed in the range $2 \leq x_c \leq 8$, $1 \leq y_c \leq 9$, $x_c + 1.5 \leq x_g \leq 10$, $0 \leq y_g \leq 10$. The start position $\boldsymbol{s} = (x_s, y_s)$ was always the same.

PRO used reward functions of the form $R_o = \exp(-\beta o)$, where $\beta = 20$ was empirically determined by observing the trajectory distributions learned by PRO for a few values of β: 1, 10, 20, 30, 40 and 50. In this problem, three objectives need to be minimized: the distance to the start position $o_1 = ||\boldsymbol{\tau}(0) - \boldsymbol{s}||$, the minimal distance to the center of the hole in the wall $o_2 = \min_t ||\boldsymbol{\tau}(t) - \boldsymbol{c}||$ and the distance to the end goal $o_3 = ||\boldsymbol{\tau}(T) - \boldsymbol{g}||$. The term $\boldsymbol{\tau}(t)$ represents the position along trajectory $\boldsymbol{\tau}$ at time step t, $\boldsymbol{\tau}(0)$ is the first position and $\boldsymbol{\tau}(T)$ is the last position.

After the initialization with the demonstrations, the self-improvement loop (Fig. 6.5) was repeated 1000 times, each time with a new random environment. Each PRO optimization took at most 200 iterations (less if convergence was achieved sooner) and used 100 trajectory samples per iteration. The kernel of the GPs used in this problem had parameter $\alpha = 10^{-3}$. This value was empirically determined by trying a few different powers of 10 and observing the GP inferences. During test, our learning system can compute ProMPs on the fly, solving the task in a dynamic environment (See Fig. 6.7).

In order to verify if the performance observed during test was due to the self-optimization procedure or simply due to the human demonstrations, we have performed the comparison depicted in Fig. 6.8. Note that by simply applying Gaussian Process Regression based on the demonstrations to infer ProMPs given a new random scenario leads to high errors with respect to the desired start and end distances as well as negative signed Euclidean distances indicating collisions with the walls. On the other hand, the self-improvement procedure involving PRO gradually leads to a better mapping from environments to ProMPs, which is evidenced by smaller errors with respect to the desired start and end positions as well as higher signed Euclidean distances to the obstacles.

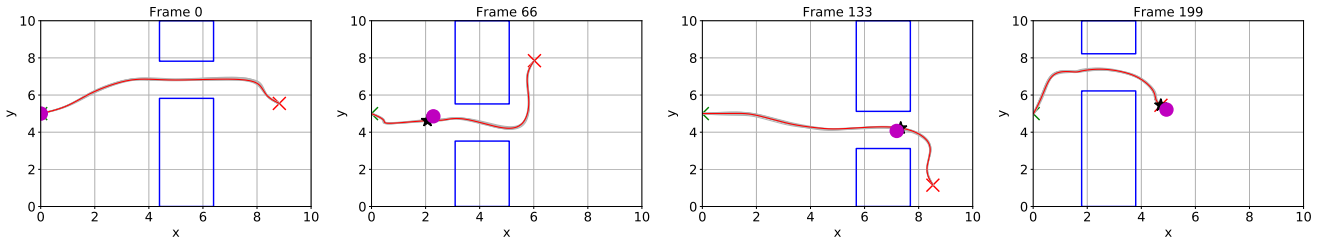


Figure 6.7.: The magenta point particle has to move from the start position (green ×) to the target position (red ×) without hitting the walls (blue rectangles). The position of the hole in the wall and the target position change with time. As these positions change, our learning system computes the corresponding trajectory distributions on the fly to solve this task. The red line corresponds to the mean of the computed trajectory distribution. The light gray trajectories are samples from the computed distribution. The black star-shaped marker moves forward along the mean of the current distribution. The magenta point particle tracks the black star-shaped marker with a PD controller. This figure depicts four frames of a test case.

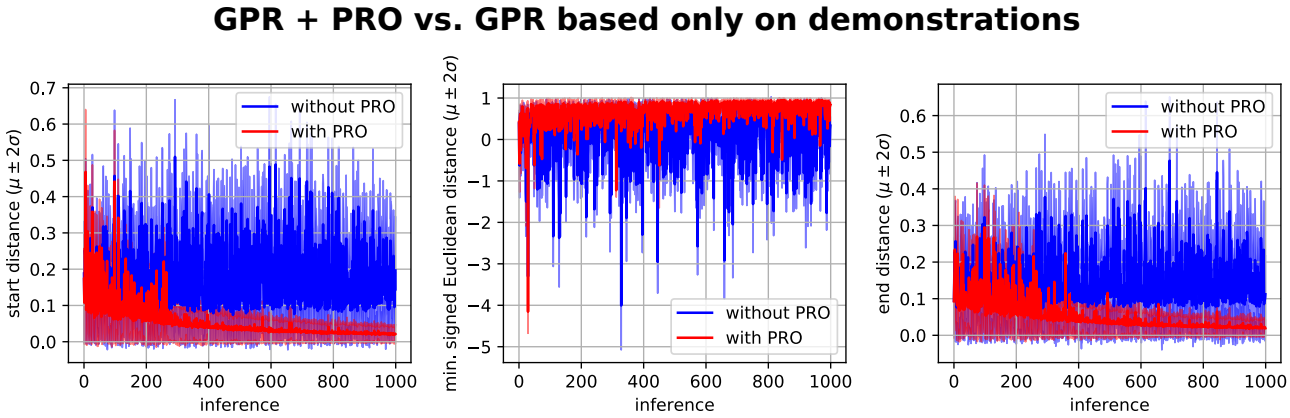
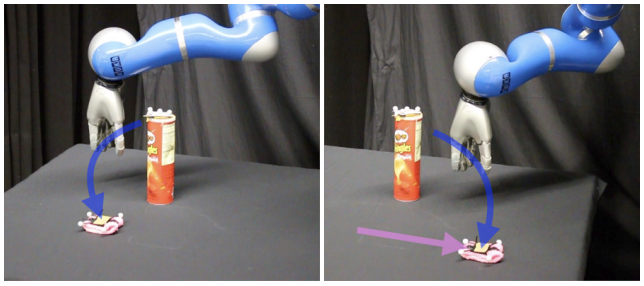


Figure 6.8.: Gaussian Process regression (GPR) plus PRO versus GPR based only on the human demonstrations without refinement by PRO. These learning curves refer to the problem depicted in Fig. 6.7. The algorithm using GPR + PRO used GPR to compute a ProMP given a random environment, optimized the inferred ProMP by using PRO, updated its dataset of environments and corresponding ProMPs and repeated this procedure for a new random environment. The algorithm using only GPR did not optimize the GPR inferences. In this case, the dataset of environments and corresponding ProMPs was solely based on the demonstrations. Both algorithms performed 1000 inferences of ProMPs given the same random environments. These plots show the performance measurements for the ProMPs inferred by GPR before PRO optimization for each of the 1000 random environments. We conclude that GPR based only on the demonstrations does not generalize well to new environments as our proposed method, which uses PRO. PRO gradually improves the mapping from environments to ProMPs.

6.5.3 Adaptation in Dynamic Environments — Robot Arm

The problem addressed in this section is depicted in Figs. 6.1 and 6.9. The obstacle (Pringles can) and the target (pink object) are tracked by using a motion capture system (OptiTrack). First, a human provides demonstrations by moving the 7-DoF robot arm in gravity compensation mode. Demonstrations were provided for 20 different environment configurations. There were three demonstrations for each environment. The configurations were determined by arbitrarily choosing positions on the table for the obstacle and the target. The start position for the robot arm was always the same.



(a) Around the obstacle from the right (b) Around the obstacle from the left

Figure 6.9.: Our learning system infers a distribution of trajectories (ProMP) given the current state of the environment. An inverse-dynamics based feedback controller tracks the mean of the inferred distribution. **(a)** The robot goes around the obstacle (a Pringles can) from the right side to reach the target (a pink object). **(b)** Given that the user changes the position of the target while the robot is moving, the robot switches on the fly to another ProMP, going around the obstacle from the left side.

Each environment in this problem was represented by the vector $\boldsymbol{e} = \left[x_p, y_p, x_g, y_g\right]^\top$, where (x_p, y_p) was the position of the Pringles can and (x_g, y_g) was the end goal position.

We have noticed that initializing our learning system only with the human demonstrations for 20 different situations was not enough to learn a mapping capable of dealing with any obstacle and target positions on the table. For this reason, we have decided to extend the set of demonstrations with ProMPs based on prior knowledge. These ProMPs had mean trajectory going directly from the start position to the target position irrespective of the obstacle position and the variance of each ProMP weight was the average variance for that weight based on the demonstrations. The GPs were thus initialized with ProMPs for 2023 different environments (including environments for which human demonstrations were given). The additional 2003 environments were generated by taking obstacle and target positions of a grid inside a range of possible positions delimited by the corners of the table and excluding configurations with the obstacle and the target too close to each other.

PRO used reward functions of the form $R_o = \exp(-\beta o)$, where $\beta = 200$ was empirically determined by observing the trajectory distributions learned by PRO for a few values of β: 1, 10, 20, 30, 40, 50, 100, 200 and 300. In this problem, three objectives need to be minimized: $o_1 = ||\boldsymbol{\tau}(0) - \boldsymbol{s}|| + d$, $o_2 = \max(-\min_t ||\boldsymbol{\tau}(t) - \boldsymbol{p}||, -0.2) + d$ and $o_3 = ||\boldsymbol{\tau}(T) - \boldsymbol{g}|| + d$.

The term $d = \frac{1}{T}\sum_{t=0}^{T} ||\boldsymbol{\tau}(t) - \boldsymbol{\tau}_{\text{direct}}(t)||$ is the average distance to the direct path $\boldsymbol{\tau}_{\text{direct}}$ from the start to the end goal. This term was added to each of the objectives to avoid large deviations from the direct path to the end goal. Apart of this term, o_1 and o_3 are very similar to objectives described in Section 6.5.2. The variable $\boldsymbol{p} = \left(x_p, y_p\right)$ in o_2 is the position of the Pringles can. Minimizing o_2 has the effect of avoiding the Pringles can without going too far away from it because distances to the Pringles can larger than 20 cm do not result in additional reward.

The self-improvement loop (Fig. 6.5) was repeated 2023 times (one time for each environment in the initialization data set). Each PRO optimization took at most 50 iterations (less if convergence was achieved sooner) and used 100 trajectory samples per iteration. The kernel of the GPs used in this problem had parameter $\alpha = 1$. This value was empirically determined by trying a few different powers of 10 and observing the GP inferences. During test, the robot is able to successfully execute the reaching task even when the human moves the obstacle or the target while the robot is moving (See Figs. 6.1 and 6.9).

6.6 Epilogue

This chapter presented a new algorithm to optimize trajectory distributions, PRO. In this algorithm, the concept of Pearson correlation is used to determine the relevance of each trajectory parameter to each optimization objective. Moreover, a framework which uses PRO and GP regression has been presented, which is able to compute trajectory distributions on the fly to solve tasks in dynamic environments. PRO can be used to optimize upon suboptimal demonstrated trajectories. An application of PRO to assisted teleoperation has been demonstrated. Our full framework is able to solve planning problems in dynamic environments in an experiment involving a point particle and in a real robot experiment with a 7-DoF robot arm.

The next step in this research is to apply our framework to assisted teleoperation in dynamic environments. We will also investigate other reward functions for the assisted teleoperation task to avoid the necessity of intermediate via points and the separation in two optimization problems.
In this work, we have used Gaussian Processes with a fixed covariance function instead of addressing the model selection problem. In our case, it is a difficult problem because we do not have a supervised learning setup. The ProMPs corresponding to each new environment configuration are output by PRO, which itself gets initialized by the inferences produced by Gaussian Process Regression. As future work, it would be thus interesting to investigate to which extent model selection can be helpful in our setup.
A limitation of PRO is that it learns ProMPs with diagonal covariance matrices instead of full covariance matrices. In the problem depicted in Fig. 6.3, if the red line had a certain slope, the solution of PRO would converge to a dot just as RWR, instead of preserving the original variance along the line. In the future, we intend to investigate the practical implications of this limitation and look for ways to learn a full covariance matrix while applying the concept of relevance functions.

7 Assisted Teleoperation in Changing Environments with a Mixture of Virtual Guides

Haptic guidance is a powerful technique to combine the strengths of humans and autonomous systems for teleoperation. The autonomous system can provide haptic cues to enable the operator to perform precise movements; the operator can interfere with the plan of the autonomous system leveraging his/her superior cognitive capabilities. However, providing haptic cues such that the individual strengths are not impaired is challenging because low forces provide little guidance, whereas strong forces can hinder the operator in realizing his/her plan. Based on variational inference, we learn a Gaussian mixture model (GMM) over trajectories to accomplish a given task. The learned GMM is used to construct a potential field which determines the haptic cues. The potential field smoothly changes during teleoperation based on our updated belief over the plans and their respective phases. Furthermore, we can learn new plans online when the operator does not follow any of the proposed plans, or after changes in the environment. User studies confirm that our framework helps users perform teleoperation tasks more accurately than without haptic cues and, in some cases, faster. Moreover, we demonstrate the use of our framework to help a subject teleoperate a 7-DoF manipulator in a pick-and-place task.

7.1 Prologue

Robots can perform precise and fast movements and can exert large forces on their environment, which makes them powerful tools for solving a variety of tasks that would be very laborious for human workers. Furthermore, they can be deployed in hazardous environments, e.g., for sorting nuclear waste [93]. However, especially when operating in unstructured environments, fully autonomous robots may not provide the required reliability. On the other hand, manually controlling the robots, for example via teleoperation, is often cumbersome and inefficient. Hence, there is great interest in shared autonomy, where both, the human operator and the autonomous system, influence the robot's actions [94].

In safety-critical environments, it is often desirable that the operator remains in full control of the robot's movements. For example, by teleoperation of a robot with a master haptic device [95], the autonomous system should apply a wrench on the end-effector handle that is large enough to provide guidance, yet weaker than the operator's wrench. However, even if the operator can apply higher wrenches on the handle than the autonomous system, the haptic guidance can still be inconvenient, especially if the plans of both agents are badly aligned, or if the haptic guidance changes suddenly.

Figure 7.1.: User operating the haptic device Haption Virtuose 6D to perform teleoperation tasks.

A way of avoiding a misalignment between the plans of the human and of the autonomous system is to vary the stiffness of the master device according to the required precision at each state. This way, the user can easily deviate from the plan proposed by the autonomous system when he/she sees fit as long as this deviation does not compromise the necessary accuracy to solve the task. For example, the robot may need to be controlled with high precision while picking up a small item, whereas the precision while moving towards that item can be rather low since the intention of the human might be still unclear. An autonomous system that takes into account these variabilities while solving a task could apply relatively high wrenches while picking up the small item and restrict itself to merely providing cues while guiding the operator towards that object, enabling the operator to move towards another object for example. In order to encode these variabilities, probabilistic trajectory representations are often used for providing the haptic guidance [84, 94].

Gaussian distributions over trajectories, such as probabilistic movement primitives (ProMPs) [35], are particularly convenient because their mean can correspond to a reference trajectory and their covariance matrix can encode the variability along that trajectory. However, a single ProMP only encodes a single (noisy) plan and may thus fail to provide adequate haptic feedback if several plans are feasible. For example, the planned trajectory might avoid an obstacle from the left, whereas the operator may prefer to avoid it from the right. As both agents would try to enforce their plan, the operator would need to apply large wrenches to the handle of the haptic device to override the plan of the autonomous system.

Instead, the autonomous system should consider several possible plans and should, furthermore, be aware of the fact that none of its plans might be in accordance with the operator's intentions.
We address these challenges by providing haptic feedback based on a mixture of ProMPs, where one of the ProMPs is fixed and has very high variance for all time steps. This ProMP corresponds to a "freelance" plan which induces negligible wrenches. The remaining ProMPs are the mixture components of a GMM over trajectories which is learned in an episodic maximum entropy reinforcement learning setting. In this work, a method for variational inference, VIPS [96], is used to learn this Gaussian mixture model.
We update our belief over the plan that the operator is following, as well as its phase, based on the operator's actions. If the operator does not seem to follow any of the plans of the autonomous system, our framework naturally assigns a high probability to the freelance plan and the operator will receive low haptic feedback. In that case, we can plan a new trajectory online and smoothly blend it in. Furthermore, plans that get invalidated due to changes in the environment can be detected and removed.
In order to provide haptic feedback based on our current belief of the operator's intention, we construct a potential field where each discretized phase of each plan is represented by the energy of a Gaussian distribution in end-effector space and use its gradient to provide haptic feedback. The potential field is thus given by the negative log probability density of a Gaussian mixture model, where the component's weights are given by our belief, and where each plan creates a valley along which the operator is guided.
The efficacy of the proposed method is validated by experiments in which users perform teleoperation tasks by manipulating a haptic device, the Haption Virtuose 6D (see Fig. 7.1). One of the tasks consists of teleoperating a 7-DoF robot arm to pick up objects. The human can switch between different assistive guides and our system helps the user pick up objects accurately. The second task consists of translating and rotating a pole in a virtual environment to reach a certain pose while avoiding obstacles. Both tasks have multiple possible solutions, which are learned by our method.

7.2 Learning a Mixture of ProMPs

In order to encode the task variabilities, we want to represent each plan by a trajectory distribution with phase-dependent variance as illustrated in Fig. 7.2. We therefore represent each plan as a Probabilistic Movement Primitive (ProMP) [35]. A ProMP is a Gaussian distribution of trajectories. This representation is convenient for our needs since the variance of this Gaussian can be used to modulate the stiffness of an autonomous system in a shared autonomy setting. Moreover, ProMPs provide a compact representation of trajectories, which is helpful for reinforcement learning algorithms.
ProMPs represent a trajectory of end-effector poses $\mathbf{x}(\nu)$ as a function $\mathbf{x}_\mathbf{w}(\nu) = \mathbf{\Phi}(\nu)^\top \mathbf{w}$ that is linear in radial basis functions. These basis functions are usually equally spaced along the movement phase $\nu \in [0, 1]$. The matrix $\mathbf{\Phi}(\nu)$ is a block-diagonal matrix of radial basis functions, such that every weight only affects a single dimension. Since the pose $\mathbf{x}_\mathbf{w}(\nu)$ is an affine function of the weights $\mathbf{w}$, a Gaussian distribution over weights $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_w, \mathbf{\Sigma}_w)$ induces a Gaussian distribution over poses

$$p(\mathbf{x}|\nu) = \mathcal{N}\left(\mathbf{x}|\mathbf{\Phi}(\nu)^\top \boldsymbol{\mu}_w, \mathbf{\Phi}(\nu)^\top \mathbf{\Sigma}_w \mathbf{\Phi}(\nu)\right).$$

Hence, a mixture of ProMPs can be represented by a Gaussian mixture model over weights

$$p(\mathbf{w}) = \sum_{o=1}^{N_o} p(o)p(\mathbf{w}|o),$$

where a multinomial distribution $p(o)$ assigns a normalized weight to each option $p(\mathbf{w}|o) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_o, \mathbf{\Sigma}_o)$ and N_o is the number of options.

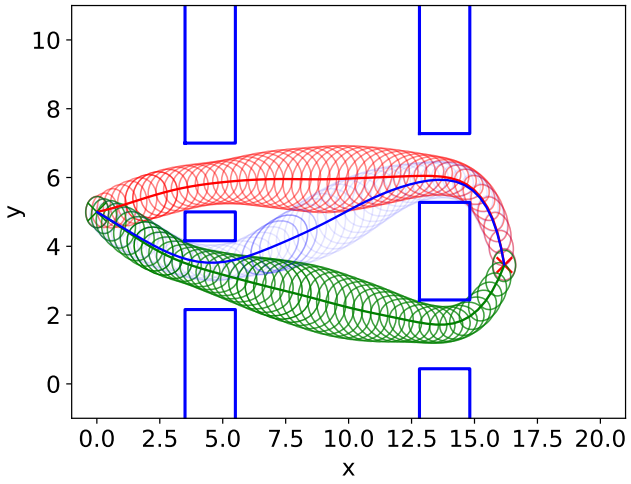


Figure 7.2.: Mixture of three trajectory distributions (ProMPs) learned in a 2D scenario. The blue lines represent the sides of walls. A point particle needs to move from a start position (green ×) to an end position (red ×) while avoiding the walls. VIPS can learn a mixture of ProMPs to solve this task. The ellipses represent Gaussians in Cartesian space, which are used to construct a potential field. This potential field is used to give the user haptic cues as explained in detail in Section 7.3.

We consider a reinforcement learning setting with a reward function $r(\mathbf{x}_\mathbf{w}(\nu), \nu)$ that depends on the end-effector pose $\mathbf{x}$ and the phase ν. We aim to learn a mixture of ProMPs that achieves high episodic reward $r(\mathbf{w}) = \int_0^1 r(\mathbf{x}_\mathbf{w}(\nu), \nu) d\nu$ in expectation while maintaining high entropy. These assumptions can be phrased as the optimization problem

$$\arg\max_{p(o),\boldsymbol{\mu}_o,\boldsymbol{\Sigma}_o} \sum_{o=1}^{N_o} p(o) \int_\mathbf{w} p(\mathbf{w}|o) r(\mathbf{w}) d\mathbf{w} + H(p(\mathbf{w})) \tag{7.1}$$

with the Shannon entropy of the Gaussian mixture model $H(p(\mathbf{w}))$. The entropy objective rewards variability and is crucial to prevent preliminary convergence to a single trajectory. Please note that scaling the reward function affects the optimal solution due to the entropy objective. We assume that the reward function is adequately scaled to result in sufficient variability without inducing undesirable trajectories.
As we consider an episodic setting, that is, we do not assume nor exploit access to time-series data, the maximum entropy reinforcement learning problem (Eq. 7.1) can be equivalently framed as variational inference. The connection between this reinforcement learning formulation and variational inference can be shown by introducing a reward distribution

$$p_r(\mathbf{w}) = \frac{1}{Z_r} \exp(r(\mathbf{w}))$$

with partition function $Z_r = \int_\mathbf{w} p_r(\mathbf{w}) d\mathbf{w}$. The optimization problem (Eq. 7.1) can now be reformulated as the variational inference problem of approximating the reward distribution $p_r(\mathbf{w})$ with a Gaussian mixture model $p(\mathbf{w})$ by minimizing the reverse Kullback-Leibler divergence $D_{\mathrm{KL}}(p(\mathbf{w})||p_r(\mathbf{w}))$. This optimization problem is

$$\begin{aligned}
&\arg\min_{p(o),\boldsymbol{\mu}_o,\boldsymbol{\Sigma}_o} D_{\mathrm{KL}}(p(\mathbf{w})||p_r(\mathbf{w})) \\
&= \arg\max_{p(o),\boldsymbol{\mu}_o,\boldsymbol{\Sigma}_o} \int_\mathbf{w} p(\mathbf{w}) \log p_r(\mathbf{w}) d\mathbf{w} + H(p(\mathbf{w})) \\
&= \arg\max_{p(o),\boldsymbol{\mu}_o,\boldsymbol{\Sigma}_o} \int_\mathbf{w} p(\mathbf{w}) r(\mathbf{w}) d\mathbf{w} - \log Z_r + H(p(\mathbf{w})).
\end{aligned} \tag{7.2}$$

As the constant partition function Z_r does not affect the solution of the optimization problem, the Optimization Problems (7.1) and (7.2) are equivalent. Therefore, we can learn a mixture of ProMPs with any variational inference method that is capable of learning multi-modal Gaussian mixture models. We use variational inference by policy search (VIPS) [96], a recent variational inference method that focuses on GMMs. Fig. 7.2 shows a mixture of three ProMPs learned by VIPS to solve a planning problem in a 2D scenario.

7.3 Computing the Haptic Cues

Haptic cues to the operator are provided by constructing a potential field in end-effector space based on our Gaussian mixture model in weight space. More precisely, we discretize the phase ν in T_o steps for each plan, denoted by $\nu_1, \dots \nu_{T_o}$ and compute the marginal distribution

$$\begin{aligned}
p(\mathbf{x}) &= \sum_{o=1}^{N_o} p(o) \sum_{i=1}^{T_o} p(\nu_i|o) p(\mathbf{x}|\nu_i, o) \\
&= \sum_{o=1}^{N_o} p(o) \sum_{i=1}^{T_o} p(\nu_i|o) \mathcal{N}(\mathbf{x}|\boldsymbol{\Phi}(\nu_i)^\top \boldsymbol{\mu}_o, \boldsymbol{\Phi}(\nu_i)^\top \boldsymbol{\Sigma}_o \boldsymbol{\Phi}(\nu_i)) \\
&= \sum_{o=1}^{N_o} p(o) \sum_{i=1}^{T_o} p(\nu_i|o) \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{o,i}, \boldsymbol{\Sigma}_{o,i})
\end{aligned}$$

using our belief $p(\nu_i|o)$ about the current phase of each plan o and the distribution $p(\mathbf{x}|\nu_i, o)$ over end-effector poses at phase ν_i in plan o. We will discuss in Section 7.4 how we adapt online the belief $p(\nu_i|o)$ over the phase and also the belief $p(o)$ over the plan.
The marginal distribution $p(\mathbf{x})$ corresponds to a Gaussian mixture model that contains Gaussian components for each plan and each phase. Similarly to Hamiltonian MCMC [47], we define a potential energy $E(\mathbf{x}) = -\log p(\mathbf{x})$ based on the negative log of the marginal, which induces a wrench that is given by the negative gradient of the energy

$$\boldsymbol{\tau}(\mathbf{x}) = -\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial \log p(\mathbf{x})}{\partial \mathbf{x}}.$$

By applying the wrench $\boldsymbol{\tau}(\mathbf{x})$, we guide the operator towards low energy regions based on the marginal $p(\mathbf{x})$. As the marginal is computed based on an approximation of the episodic reward distribution $p_r(\mathbf{w})$, the operator is guided towards trajectories that provide high episodic reward. Please note that directly using the negative reward function $-r(\mathbf{x}, \nu)$ for constructing a potential field would guide the operator towards regions of high immediate reward, whereas our learned, planning-based potential field assists the operator in achieving high episodic reward. For example, consider a goal position that is directly behind a wall and a reward function that penalizes the distance to the goal and close proximity to the wall. Whereas the immediate reward results in a potential field that would keep the operator in the current position, our learned potential field would create several valleys along which the operator is guided around the wall. Furthermore, the energy of the marginal distribution $p(\mathbf{x})$ corresponds to the negative log of a Gaussian mixture model and is thus differentiable and smooth. A smooth potential field is highly desirable in order to avoid sudden jumps in the direction or magnitude of the wrench that is applied to the handle of the master device. The gradient of the log of the marginal is

$$\frac{\partial \log p(\mathbf{x})}{\partial \mathbf{x}} = \sum_{o=1}^{N_o} \sum_{i=1}^{T_o} p(o, \nu_i|\mathbf{x}) \Sigma_{o,i}^{-1} (\mu_{o,i} - \mathbf{x}). \tag{7.3}$$

Hence, each component attracts the operator with a force of magnitude proportional to the distance to its mean and weighted by its responsibility

$$p(o, \nu_i|\mathbf{x}) = \frac{p(o)p(\nu_i|o)p(\mathbf{x}|\nu_i, o)}{p(\mathbf{x})}.$$

The operator would typically get small haptic cues if he/she is close to one of the components because, due to the proximity to that component, its responsibility would often be close to one and its force contribution would be close to zero. However, if the operator is far from every component, all components would have large force contributions and their weighted average would typically also be large. Such behavior can be desirable if we do not want to allow the operator to leave the planned trajectories.

However, if the operator should keep full control, such behavior would be undesirable even if the maximum wrench gets capped. In such cases, we can add an additional component with large variance to our mixture model that covers the whole workspace. Such a component corresponds to a freelance plan with a single phase, where the operator can move freely within the workspace. Due to its high variance, this component always has low force contributions and gets low responsibility if the operator is close to the planning based components, but high responsibility if the operator is far from any planned components.

Without the user, when assuming the beliefs $p(o)$ and $p(\nu_i|o)$ to be fixed and the control rate at which the haptic cues are updated to be infinite, no energy gets injected into our system. We include a damping term, such that the total wrench is

$$\boldsymbol{\tau}_{\text{total}}(\mathbf{x}, \dot{\mathbf{x}}) = \boldsymbol{\tau}(\mathbf{x}) - k_{\text{damp}}\dot{\mathbf{x}}. \tag{7.4}$$

By including such a damping, the total energy of the system always decreases, which leads to its stability. The damping term also prevents oscillatory force feedback close to low-variance components that were caused by our limited control rate.

7.4 Adapting the Mixture Weights by Updating the Belief

In order to assist the user in a teleoperation task, our system needs to infer from the pose of the handle of the haptic device commanded by the human which ProMP (also referred to as option or plan) the human intends to follow. We may also want to guide the human forward along the intended plan, while enabling the user to stop or move backward if he/she wants. Therefore, we may also need to infer the intended phase along the plan of interest.

These inference problems are formalized by using the weights $p(o)$ and $p(\nu|o)$ to reflect our belief about the plan o that the operator is currently employing and their respective phases ν, respectively. We initialize $p(o)$ based on the weights that were learned by VIPS, which assigns higher weights to components that achieve higher expected reward, higher entropy or that differ from the remaining components. We initialize our belief $p(\nu|o)$ over the phase as a uniform distribution.

In order to update our belief during teleoperation, we formulate a Hidden Markov Model (HMM) where the true plan o_t of the operator and the phases $\nu_t(o)$ at time t are hidden variables. Please note that t corresponds to the discretized real time that increments at the control rate at which we send the desired wrench to the haptic device. Fig. 7.3 represents this HMM. The transition probabilities $p(\nu_{t+1}(o)|\nu_t(o))$ and $p(o_{t+1}|o_t)$ are explained in Sections 7.4.1 and 7.4.2, respectively. The emission probability $p_{\text{obs}}(\mathbf{x}_t|\nu_t(o), o_t)$ is obtained by rescaling the Covariance matrix of $p(\mathbf{x}_t|\nu_t(o), o_t)$ to allow for a larger uncertainty in the observation $\mathbf{x}_t$. The rescaling factor for the Covariance matrix is chosen such that the belief about the plan $p(o)$ does not converge too fast in the face of new observations.

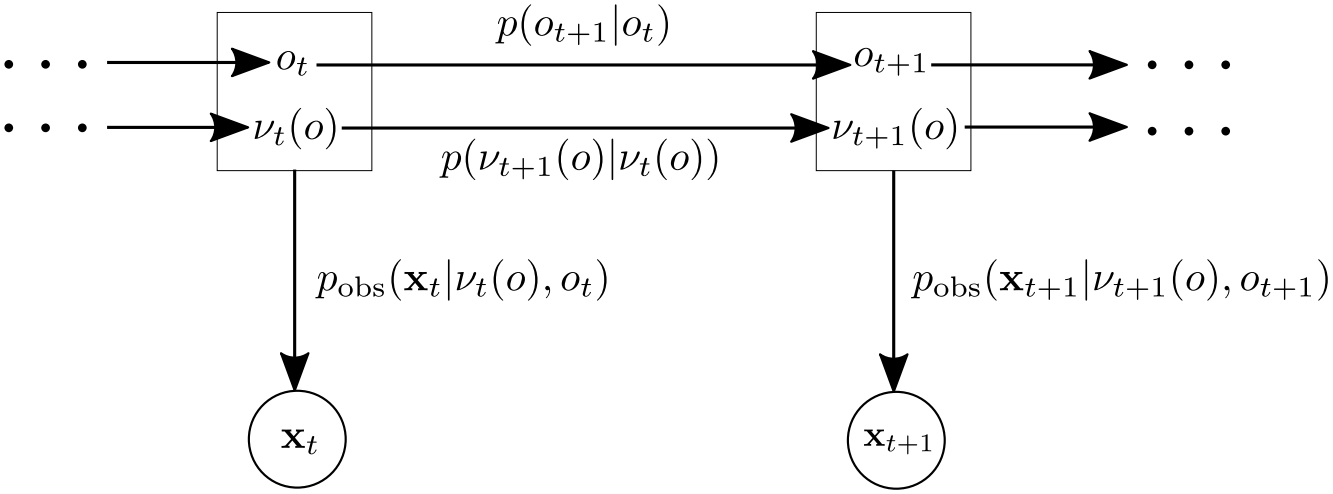


Figure 7.3.: Hidden Markov Model (HMM) underlying the update of the belief about the plan o and phase ν.

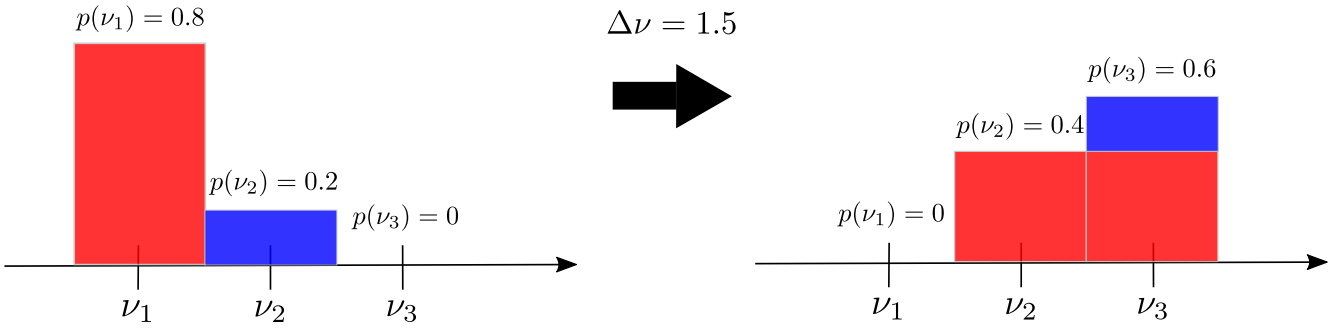


Figure 7.4.: Example of change in the probability distribution $p(\nu)$ induced by the shifting operator $\mathcal{T}_{\Delta\nu}(p)$ when the phase is discretized in only three possible values for simplification. With $\Delta\nu = 1.5$, half of the probability mass of ν_1 gets assigned to ν_2 and the other half gets assigned to ν_3. Because ν_3 is the last phase, all the probability mass of ν_2 gets shifted to ν_3.

7.4.1 Updating the Belief about the Phase

For modeling the transition probability of the phase $p(\nu_{t+1}(o)|\nu_t(o))$, we assume that the phase progresses by $\Delta\nu$ with probability p_{progress} and changes to a random phase otherwise. A prior belief about the phase at time step t can thus be computed based on the posterior belief at time step $t-1$,

$$p_{\text{prior,t}}(\nu|o) = p_{\text{progress}}\mathcal{T}_{\Delta\nu}\big(p_{\text{post,t-1}}(\nu|o)\big) + (1 - p_{\text{progress}})\frac{1}{T_o}, \tag{7.5}$$

where the shifting operator $\mathcal{T}_{\Delta\nu}(p)$ shifts the probability mass of the distribution p by $\Delta\nu$ forwards in time whereby no mass gets shifted beyond the last phase ν_{T_o}. Please note that $\Delta\nu$ can be any positive real value by shifting fractions of the probability mass of the distribution p. For example, when shifting by $\Delta\nu = 1.5$, half of the probability at ν_1 would get assigned to ν_2 and the other half would get assigned to ν_3. Fig. 7.4 presents a visualization of the effects of the shifting operator $\mathcal{T}_{\Delta\nu}(p)$ for a case where $T_o = 3$.
Based on recursive Bayesian estimation, the posterior belief at time t is

$$p_{\text{t,post}}(\nu|o) \propto p_{\text{t,prior}}(\nu|o)p_{\text{obs}}(\mathbf{x}_t|\nu, o).$$

However, when providing haptic cues based on the posterior belief of the phase at time step t, we would not provide any incentive to the operator to progress with the plan. Instead, we compute the haptic cues based on the prior belief of the phase at the next time step, $p_{\text{prior,t+1}}(\nu|o)$. As we assume the phase to progress by $\Delta\nu$ at the control rate, computing the negative energy gradient (Eq. 7.3) based on the prior belief of the next time step, results into a wrench that tends to pull the operator towards the upcoming components for each plan, providing an incentive to progress.

7.4.2 Updating the Belief about the Plan

We model the transition probability of the plan as

$$p(o|o') = \begin{cases} (1 - p_{\text{switch}}), \text{ if } o' = o \\ p_{\text{switch}}/(N_o - 1), \text{ otherwise} \end{cases} \tag{7.6}$$

with a small probability p_{switch} of changing the plan and N_o corresponding to the number of plans. We therefore assume that the operator likely follows the same plan as during the last time step, but with low probability might change to another random plan. The Bayesian belief update

$$p_{\text{post,t}}(o) \propto p_{\text{prior,t}}(o) p(\mathbf{x}_t|o) \tag{7.7}$$

$$\propto \int_{o'} p_{t-1}(o') p(o|o') do' \sum_{i=1}^{T_o} p_t(\nu_i|o) p_{\text{obs}}(\mathbf{x}_t|\nu_i, o) \tag{7.8}$$

corresponds to the posterior belief of the plan that is pursued by the operator at time t.

7.5 Adapting the Plans Online

The computational time required to plan a mixture of ProMPs with the proposed method depends strongly on the dimension of the weights $\mathbf{w}$, and thus on the number of degrees of freedom and basis functions, as well as on the number of components $p(\mathbf{x}|o)$ that are to be learned. VIPS [96] can learn components with full covariance matrices Σ_o; however, to speed up the online replanning of ProMPs, we only learn diagonal covariance matrices in this work, which greatly reduces the number of parameters to be learned. This simplification ignores correlations between different phases and between different degrees of freedoms, resulting in mixture components $p(\mathbf{x}_t|\nu, o)$ in the end-effector space that are axis-aligned. We found VIPS to be sufficiently efficient for learning small mixtures of ProMPs online. We consider two kinds of events that trigger online replanning, namely, replanning due to changes in the environment and replanning if the operator does not seem to pursue any of the existing plans.

Changes of the environment can be detected for example via vision and typically invalidate the previous plans. We therefore remove the previous plans and provide haptic cues only based on the freelance plan and the new plans.

We can detect that the operator is not pursuing one of the current plans, for example, based on the smallest Mahalanobis distance to each of the end-effector components $p(\mathbf{x}|\nu_i, o)$ or based on the belief $p(o_{\text{freelance}})$ of the freelance plan. If the operator does not seem to pursue one of the previous plans, we do not need to delete the previous plans. In order to avoid sudden changes in the haptic cues, we add the new plans with negligible initial weight to the mixture of ProMPs. The weight of the new plans will typically smoothly increase due to the belief updates.

7.6 Experiments

Two experiments have been performed to demonstrate applications of our approach based on VIPS and ProMPs to assist humans perform teleoperation tasks. The first experiment demonstrates that our framework can be used to assist users teleoperate a 7-DoF robot arm by manipulating a 6-DoF haptic device. The user controls the 3D Cartesian position of the end-effector of the robot to realize pick and place tasks in a changing environment. In the second experiment, users need to control through the 6-DoF haptic device the 6D pose of a pole in a virtual environment. The users need to move the pole as quickly as possible between specific positions and avoid collisions with obstacles. In this experiment, we have performed user studies to evaluate the efficacy of our framework for assisting users in teleoperation tasks. The haptic device used in these experiments is depicted in Fig. 7.1.

7.6.1 Shared Control of the 3D End-Effector Position of a 7-DoF Robot Arm

Picking up small objects through teleoperation can be a very challenging task for a human operator due to the difficulty in accurately estimating the 3D position of the objects of interest. This experiment demonstrates how our framework can be used to facilitate such a task.

In our setup, the human manipulates a 6-DoF haptic device to teleoperate a 7-DoF robot arm. The human has control over the 3D Cartesian end-effector position of the robot arm and the joint configuration of the robot is computed through inverse kinematics. The goal of the human is to pick three little balls and release them inside a basket. The environment

may change online through the introduction of an obstacle (a Pringles can). Our framework helps the user to solve this task by learning multiple plans (ProMPs) that are used to modulate the force feedback applied by the haptic device on the human. The ProMPs are relearned online whenever there are changes in the environment or when the belief of the freelance plan is larger than 0.5. Fig. 7.5 shows three snapshots of the robot and of the environment during one trial to accomplish the pick and place task. The figure shows also the virtual guides (ProMPs) that have been learned in each situation. The balls and the obstacle have been tracked by using the OptiTrack system.
In these experiments, the reward $r(\mathbf{w}) = \boldsymbol{\psi}(\mathbf{x_w})^\top \boldsymbol{\theta}$ is a weighted sum of features. The vector $\boldsymbol{\psi}(\mathbf{x_w})$ has elements such as distance to desired start pose, distance to desired end pose, log-likelihood of trajectories designed to be higher when avoiding obstacles, etc. The vector $\boldsymbol{\theta}$ has empirically determined weights for each feature. We have tried different weights until we achieved reward functions leading to ProMPs that satisfied all the criteria of the task at hand such as avoiding obstacles, starting and ending with certain poses, not producing jerky trajectories, etc. An alternative to this approach would be using Inverse Reinforcement Learning (IRL) to determine the weights [97]. The integration of IRL into our framework is an interesting topic for future research.
In this experiment, $\mathbf{x_w}(\nu_i) = [x(\nu_i), y(\nu_i), z(\nu_i)]^\top$ represents the 3D Cartesian position of the end-effector of the slave robot arm at phase ν_i. Our feature vector evaluated for trajectory $\mathbf{x}$ is

$$\boldsymbol{\psi}(\mathbf{x}) = \left[d^2_{\text{start}}, d^2_{\text{end}}, \mathscr{L}_{\text{box}}, \mathscr{L}_{\text{obstacle}}, \sum_{i=1}^{T_o} \dot{\mathbf{x}}^2(\nu_i), \sum_{i=1}^{T_o} \ddot{\mathbf{x}}^2(\nu_i), \sum_{i=1}^{T_o} \min(0.5, z(\nu_i)) \right]^\top \tag{7.9}$$

and the vector of feature weights is $\boldsymbol{\theta} = [-5000, -5000, 5000, 5000, -500, -50000, 50]^\top$.
In (7.9), d_{start} is the distance between the start position of the trajectory and the desired start position. The variable d_{end} represents the distance between the end position and its closest target. The features $\mathscr{L}_{\text{box}}$ and $\mathscr{L}_{\text{obstacle}}$ have been designed to prevent leaving the workspace in the form of a bounding box and collisions with the obstacle, respectively. The next two features prevent high velocities and accelerations. Finally, the last feature gives higher rewards for achieving high positions inside the bounding box defining the workspace of the robot. This feature induces grasping motions from the top.
The features $\mathscr{L}_{\text{box}}$ and $\mathscr{L}_{\text{obstacle}}$ are computed based on the signed Euclidean distance between the robot end-effector and the bounding box or obstacle, respectively. First, the minimum signed Euclidean distance d between trajectory $\mathbf{x}$ and the bounding box or obstacle is computed. Subsequently, the log-likelihood of that trajectory is given by

$$\mathscr{L} = \begin{cases} \log\left(\mathcal{N}\left(0; 0, \sigma^2\right)\right), \text{ if } d \geq 0 \\ \log\left(\mathcal{N}\left(d; 0, \sigma^2\right)\right), \text{ if } d < 0 \end{cases} \tag{7.10}$$

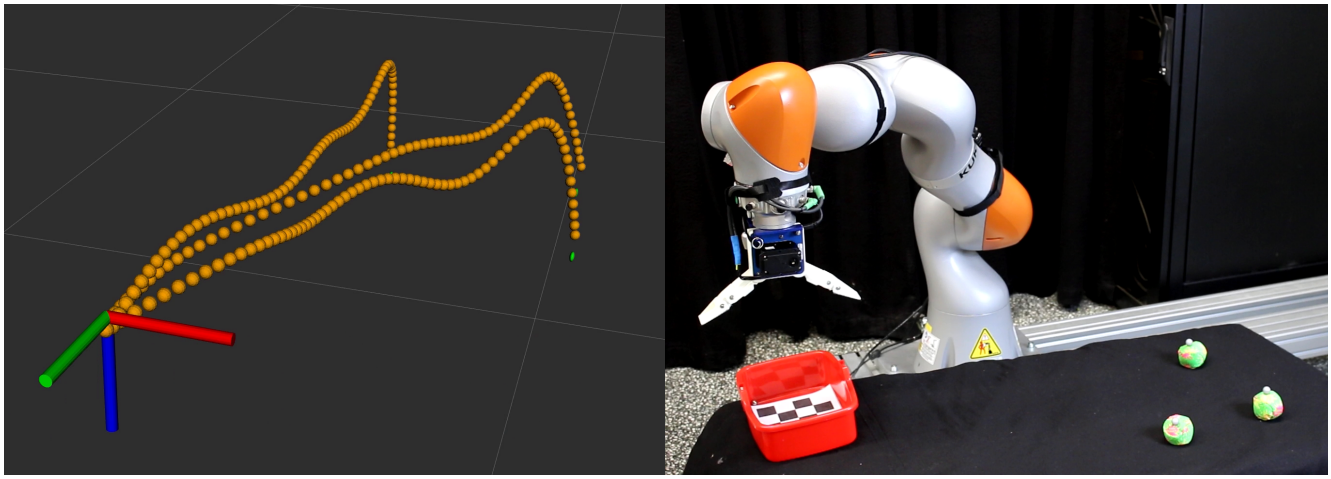
where we chose $\sigma^2 = 2$. Here, $\mathscr{L}$ represents $\mathscr{L}_{\text{box}}$ if d is the signed Euclidean distance to the bounding box or $\mathscr{L}_{\text{obstacle}}$ if d is the signed Euclidean distance to the obstacle. The signed Euclidean distance to the bounding box is positive inside the box and negative outside it. The obstacle is modeled as a cylinder. The signed Euclidean distance to the obstacle is positive outside the cylinder and negative inside it.

7.6.2 Shared Control of the 6D Pose of a Virtual Object

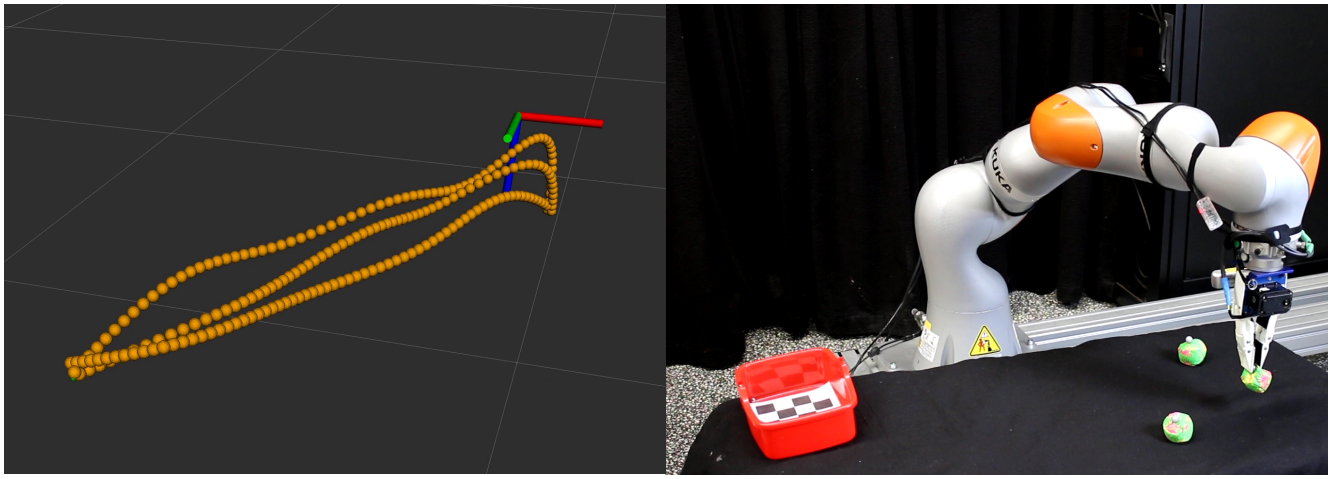
This experiment addresses the problem of assisting users in teleoperation tasks with multiple solutions involving both translations and rotations. In this case, users have to control a virtual pole by manipulating a haptic device. Our virtual environment consists of the pole, a wall with two windows, the desired start position and the desired end position (see Fig. 7.6). As in the previous experiment, the reward function has the form $r(\mathbf{w}) = \boldsymbol{\psi}(\mathbf{x_w})^\top \boldsymbol{\theta}$. This time, the feature vector for trajectory $\mathbf{x}$ is

$$\boldsymbol{\psi}(\mathbf{x}) = \left[d^2_{\text{start}}, d^2_{\text{end}}, \mathscr{L}_{\text{wall}}, \sum_{i=1}^{T_o} \dot{\mathbf{x}}^2(\nu_i), \sum_{i=1}^{T_o} \ddot{\mathbf{x}}^2(\nu_i), \sum_{i=1}^{T_o} \alpha^2(\nu_i) + \beta^2(\nu_i) + \gamma^2(\nu_i) \right]^\top \tag{7.11}$$

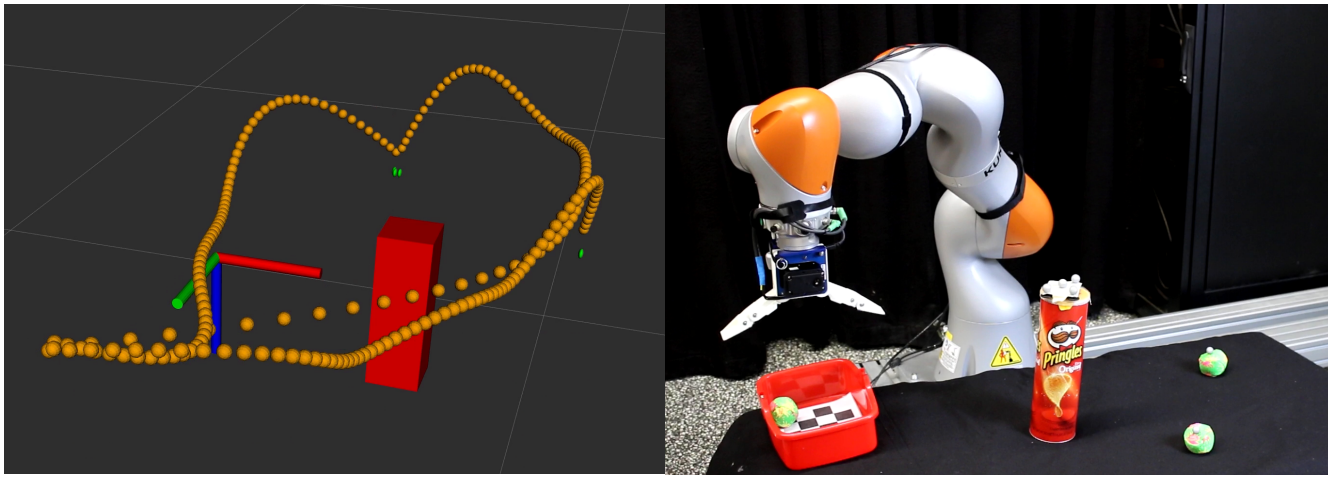
and the vector of feature weights is $\boldsymbol{\theta} = [-2.5, -5, 1000, -5, -5, -5]^\top$.
In (7.11), d_{start} is the distance between the start pose of the trajectory and the desired start pose. The variable d_{end} represents the distance between the end pose and the desired end pose. The feature $\mathscr{L}_{\text{wall}}$ has been designed to prevent collisions with the obstacle, which in this case is a wall. The next two features prevent high velocities and accelerations, both Cartesian and rotational. Finally, the last feature serves to prevent unnecessary rotations by punishing high Euler angles α, β and γ. The feature $\mathscr{L}_{\text{wall}}$ is computed according to (7.10) with d the minimum signed Euclidean distance between the pole and the wall.



(a) Initial virtual guides (left) and environment (right)



(b) Virtual guides (left) and environment (right) after first grasp



(c) Virtual guides (left) and environment (right) after introducing an obstacle

Figure 7.5.: Teleoperating a robot arm through a haptic device with force feedback. In this experiment, the user controls the Cartesian position of the robot end-effector. Our framework computes distributions of trajectories that serve as virtual guides to the user. The force feedback applied by the haptic device on the user depends on the gradient of the trajectory distributions. Our framework replans the virtual guides whenever there is a change in the environment or when the user escapes all previously planned virtual guides.

In these experiments, ten users have been requested to perform two tasks as quickly as possible while avoiding collisions with the wall. Task 1 consisted of moving the pole from the start position to the end position through any of the two windows. Task 2 consisted of moving the pole from the start position to the end position through any of the two windows and bringing it back to the start position through the other window.

Task 1 has been performed by the users with two control modes: with force feedback or without force feedback. Task 2 has been performed with three control modes: with force feedback and replanning, with force feedback and without replanning, without force feedback. The order of the control modes experienced by each user has been determined at random. When force feedback was active, our system would compute a guiding ProMP, which would be used to give the users force feedback. The initial ProMP resembles a tube starting at the start position and extending through one of the two windows until the end position. By applying a certain amount of force to the end-effector of the haptic device, users could escape the guidance of the initially planned ProMP. With replanning enabled, a new ProMP from the pole's current pose to the end pose would be planned once the user had escaped all the other previously planned ProMPs. Without replanning, no new ProMP would be planned once the user had escaped the initially planned ProMP. Fig. 7.7 shows an example of replanning enabled.

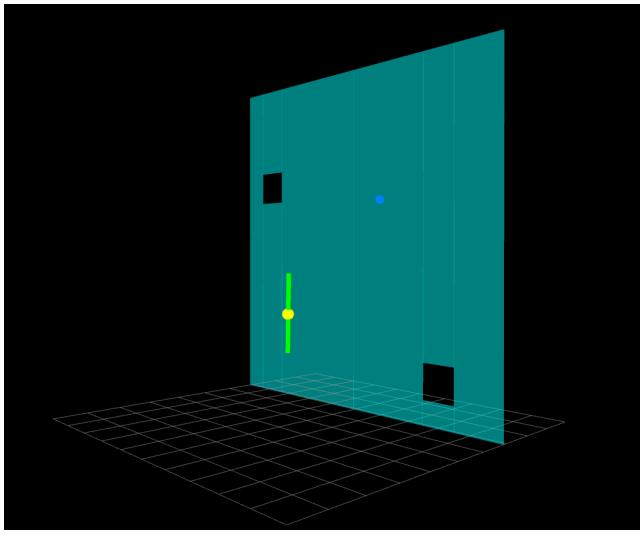


Figure 7.6.: 6D Experiment. The user operates the haptic device Haption Virtuose 6D to translate and rotate a pole in a virtual environment. The goal of the user is to move the pole from the start position (yellow sphere) to the end position (blue sphere) through one of the two windows without hitting the wall. VIPS learns a mixture of trajectory distributions (ProMPs) that helps the user achieve the necessary translations and rotations to solve this task.

In summary, each user performed Task 1 (start to end) two times (with and without force feedback) and Task 2 (round trip) three times (with force feedback and replanning, with force feedback and no replanning, without force feedback). The number of collisions with the wall in each trial has been recorded as well as the time users took to complete each task. Task 1 was considered completed once the distance between the center of the pole and the end position (marked by the blue sphere) was under a certain threshold. Task 2 was completed once, after reaching the position marked by the blue sphere, the distance between the center of the pole and the start position (marked by the yellow sphere) was under a certain threshold. Moreover, after all the trials, users have been asked "In a scale from 1 to 5, how much did you feel in control of the haptic device? 1 means completely not in control. 5 means completely in control." Fig. 7.8 shows the results.

Nonparametric ANOVA Kruskal-Wallis hypothesis tests with a significance level $\alpha = 0.05$ have been conducted to evaluate our results. For the evaluations involving more than two groups, post-hoc Conover's tests have been used to indicate which groups were significantly different.

There was a significant difference between "with force feedback" and "no force feedback" with respect to the number of collisions ($p = 0.0137$) and time to complete the task ($p = 0.0002$) in Task 1. The difference between 'with force feedback" and "no force feedback" with respect to the feeling of control reported by the users was not significant ($p = 0.9060$) in Task 1.

Kruskal-Wallis test has indicated a significant difference ($p = 0.0008$) between the three teleoperation modes in Task 2 with respect to the number of collisions. Conover's test has indicated that the differences were significant between "no force feedback" and "with force feedback, no replanning" ($p = 0.0035$) as well as between "no force feedback" and "with force feedback, with replanning" ($p = 0.0001$). Kruskal-Wallis test has indicated no significant difference ($p = 0.2050$) between the teleoperation modes in Task 2 with respect to the time users took to complete the task. There was also no significant difference ($p = 0.7563$) between the three teleoperation modes in Task 2 with respect to the feeling of control reported by the users.

Based on the box plots and hypothesis tests, we conclude that our framework helps users avoid collisions in both tasks. Furthermore, our framework helps users complete Task 1 faster than without assistance. Although the hypothesis tests did not indicate a significant difference between the teleoperation modes in Task 2 with respect to the time users took to complete the task, Fig. 7.8(b) shows two outliers for the teleoperation mode without force feedback. The modes with force feedback did not present outliers. Therefore, our framework may help users to not get lost and to not require a

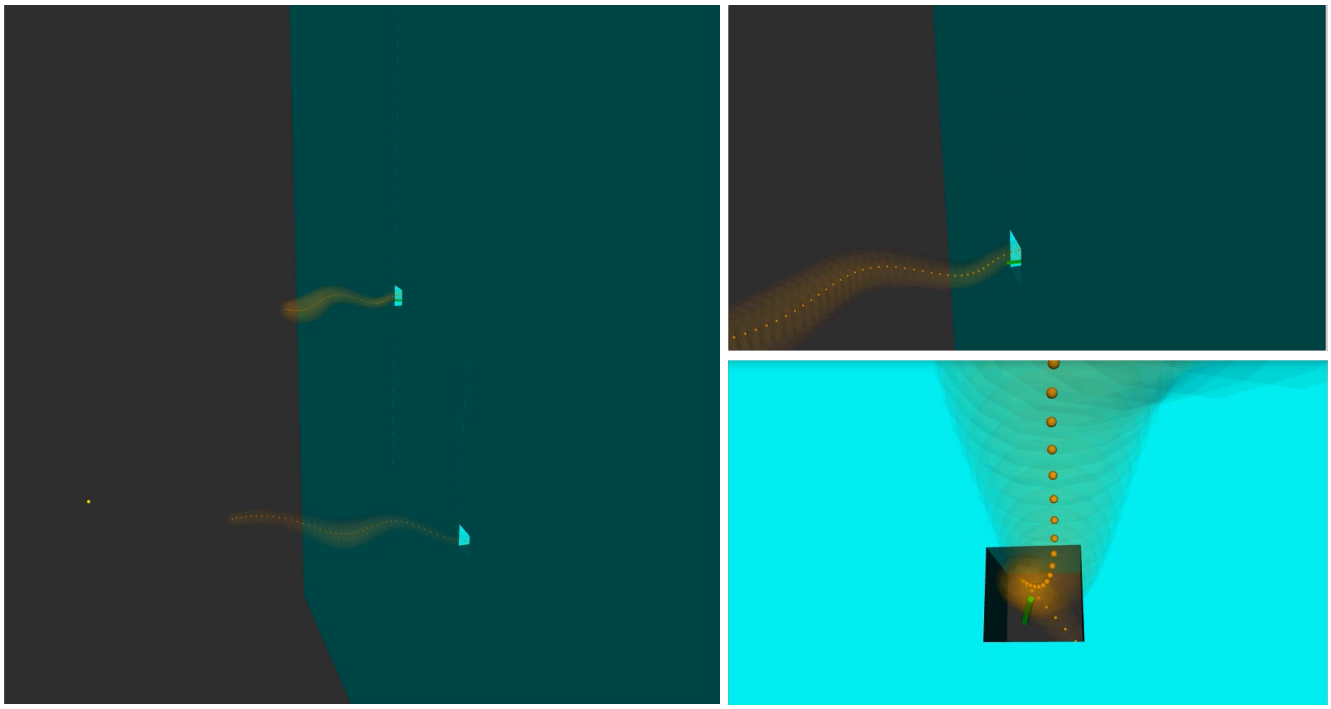


Figure 7.7.: Example of replanning in the experiment involving the teleoperation of a virtual pole through a haptic device. The little orange spheres represent positions along the mean of distributions of trajectories (ProMPs). The larger transparent orange spheres represent the variances in x, y and z corresponding to the positions depicted by the small spheres. Our framework learns ProMPs to help the user move the pole through the windows to a target position on the other side of the wall. When the user escapes the attraction of one of the ProMPs, another ProMP is learned to take the pole from its current position to the target position. The three images represent the same moment from three different perspectives.

very large amount of time to complete Task 2. Moreover, the box plots and hypothesis tests indicate that our framework helps users avoid collisions in Task 2 without compromising the time users take to complete the task.

The hypothesis tests indicate that the difference between the medians of the feeling of control reported by the users was not significant in any of the tasks. We conclude that our framework does not affect much the feeling of control of the users over the haptic device. This result is in accordance with the intended purposes of our framework, since it is supposed to assist users while giving them the freedom to diverge from the guidance of the haptic device.

7.7 Connection to Prior Work

7.7.1 Potential Fields and Dynamical Systems

Potential fields have long found applications in shared control tasks. In [98], potential fields are used to drive a robot arm away from obstacles and from the borders of the workspace as well as to let the human change the autonomous behavior of the robot. The contribution of the human to the movement can be given directly through velocities entered through a joystick or by modifying the position and the scaling factor of potential fields. The contribution of the human to the movement can be important for example if there are unknown objects in the environment or modeling errors.

Potential fields can be used to ease the teleoperation of a robot to grasp small objects. In [99], an experiment is described where a camera has been mounted on the gripper of a robot. Potential fields help the user move the gripper such that the object appears at the center of the image captured by the camera, meaning that the gripper is just above the object.

Another application of potential fields designed to help users in teleoperation tasks is shown in [100]. In that work, a control loop is proposed to assist the teleoperation of a quadrotor both in contact-free flight and when applying forces to objects. The control loop automatically slows down the quadrotor in the proximity of objects. Moreover, the user receives through the master haptic device force feedback proportional to the force applied by the quadrotor at the contact point.

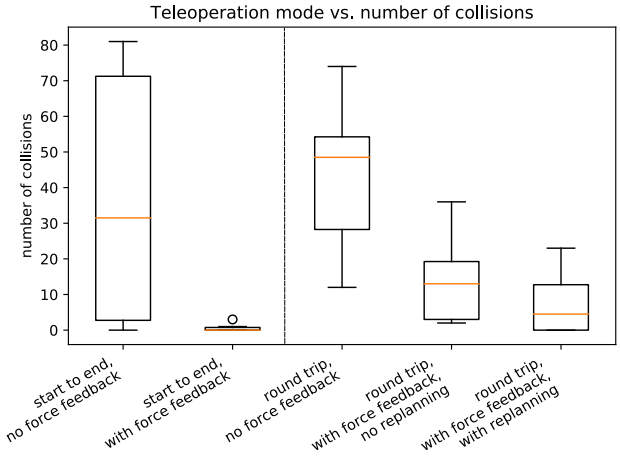
In contrast to the mentioned works, our approach consists of learning potential fields that provide the user with one or more paths to solve a task. Furthermore, our approach can deal with obstacles without getting stuck in local minima as it is often the case in classical potential fields methods.

Dynamical systems approaches more sophisticated than classical potential fields have recently been explored in tasks involving avoidance of obstacles of various shapes and interaction with humans. Huber et al. [101] have presented an approach to make the end-effector of a robot avoid convex and star-shaped obstacles while moving towards an attractor. That approach consists of multiplying the original linear dynamical system by a dynamic modulation matrix, which can be computed in closed form to guarantee the impenetrability of convex and star-shaped obstacles. Our work does not provide the same strong guarantees. On the other hand, it finds multimodal trajectory distributions to solve planning problems with arbitrary reward functions, which can be used as guiding virtual fixtures in assisted teleoperation tasks.

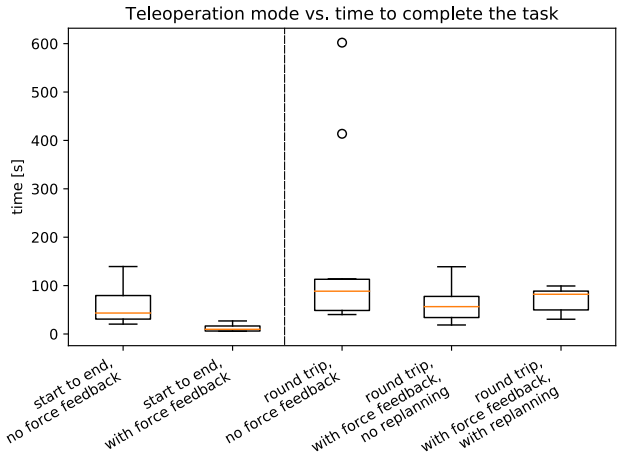
Potential fields can be learned from demonstrations, which increases the applicability of robots beyond hard-coded behaviors, and can lead to safe and reliable movements in unstructured environments. In [102], a potential energy function and a dissipative field are learned from demonstrations. The control policy is the negative gradient of the potential function minus the dissipative field. Movements with a single target are addressed. The parameters of the potential energy function and of the dissipative field such that the robot reproduces the demonstrations and converges from any start state to the target state are determined by solving two convex constrained quadratic optimization problems. Our work bears similarities to [102] in the sense that our approach also involves learning a potential field. Nevertheless, in our work, the objective is not necessarily to drive the robot along a certain path to the target. Instead, we address the problem of giving force feedback to a user in assisted teleoperation tasks. There may be several solutions to the task and the user can decide to switch from a path to another. Moreover, our potential field is learned through reinforcement learning instead of from demonstrations. This feature is especially desirable when it is hard for the human to give suitable demonstrations.

7.7.2 Gaussian Mixture Models

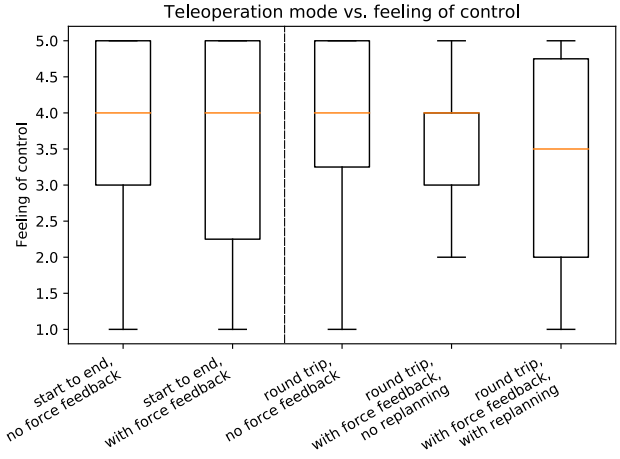
In [37], a Gaussian mixture model (GMM) of trajectories is learned from demonstrations. The learned GMM is used to model human-robot collaboration tasks. This approach is extended in [103] to enable the open-ended learning of a skill library for collaborative tasks. The open-ended learning is achieved through the use of incremental Gaussian mixture models [104], which do not require prespecifying the number of mixture components. In this chapter, a variational inference method (VIPS) to learn GMMs has been employed [96]. VIPS requires specifying a maximum number of components instead of the exact number and is used to learn GMMs given a reward function instead of demonstrations. VIPS could however in



(a) Teleoperation mode versus number of collisions



(b) Teleoperation mode versus time to complete the task



(c) Teleoperation mode versus feeling of control

Figure 7.8.: Box plots representing the data acquired in our user studies on the control of the 6D pose of a virtual pole by manipulating a haptic device.

principle been employed with an Inverse Reinforcement Learning method [97] to learn the reward function from demonstrations and the GMM to optimize that reward function.
Raiola et al. [73] also learn GMMs of trajectories from demonstrations. In their work, virtual mechanisms attract the end-effector of the robot like spring-damper systems, guiding the user in co-manipulation tasks. The GMM formulation allows for the existence of multiple virtual mechanisms. The resultant force applied to the end-effector is a combination of the forces applied by each virtual mechanism weighted by the probability of each virtual mechanism given the current end-effector position. In our work, the GMMs of trajectories are learned through reinforcement learning, which can be very helpful when the user cannot provide suitable demonstrations, e.g. in a challenging teleoperation task with several degrees of freedom.

7.7.3 Variational Inference

In [105], a method to learn a mixture of models through Variational Inference is proposed. The paper points several possible applications of this method in robotics, e.g. learning forward and inverse kinematics, planning, etc. In that work, a mixture of Gaussians is learned to cover a certain 2D space while avoiding all the obstacles in that space. Subsequently, a shortest path algorithm is used to find a sequence of mixture components from a given start position to a given end position. In our work, Variational Inference is used to learn a mixture of Gaussians over trajectories, namely, probabilistic movement primitives (ProMPs), which solve planning problems and assist users in teleoperation tasks. Covering the entire workspace with mixture components does not scale very well for higher dimensions. Therefore, in our work, the components of the learned GMM correspond directly to successful distributions of trajectories to solve the task instead of representing clusters of successful poses.

7.8 Epilogue

This chapter introduced a new approach for assisted teleoperation. A probability distribution of trajectories in the form of a Gaussian mixture model (GMM) is learned through variational inference with the VIPS algorithm. VIPS learns a GMM that maximizes an episodic reward function in expectation while maintaining high entropy. Each mixture component is a Probabilistic Movement Primitive (ProMP). The gradient of the marginal distribution in end-effector space plus a damping term is equal to the wrench applied by the haptic device on the user performing the teleoperation task. Moreover, Bayesian belief update is used to infer the plan and the phase of the movement that is intended by the user. The belief about the plan and the phase corresponds to the weights of the mixture components in end-effector space. Therefore, this belief influences the wrench applied by the haptic device on the user.
Our approach can guide the user around obstacles when classical potential fields methods could get stuck in local minima. It can learn multimodal solutions to a task and adapt online to changes in the environment or in the intention of the user, as it has been demonstrated in experiments involving the teleoperation of a 7-DoF robot arm and the control of the 6D pose of a virtual pole. User studies have demonstrated that our framework can help users perform teleoperation tasks faster and avoiding collisions with obstacles.
A promising line of research for future work would be to combine the proposed framework with Inverse Reinforcement Learning to avoid specifying the weights of the features of the reward function manually. In addition, user studies involving the teleoperation of a real robot arm would be interesting to evaluate the efficacy of our approach in more realistic scenarios. In this case, our system should also give the user haptic cues to avoid singularities and joint limits.

8 Conclusion and Future Work

There has been a large amount of research work on robots learning from humans (imitation learning) and on humans learning or getting assisted by robots. However, these two bodies of work have been mainly separated. Recent works on robot-assisted human motion and on robot-assisted human motor skill learning make significant advances in building systems to assist humans as well as in understanding human motor skill learning but they still rely on expert demonstrations or predefined trajectories. This thesis bridges the gap between both learning directions (robots learning from humans and humans learning or getting assisted by robots) by addressing both of them within a single framework based on stochastic movement representations and reinforcement learning. Our work presents contributions to solve challenges in imitation learning, improve and generalize motor skills through reinforcement learning. Furthermore, the stochastic movement models learned from expert demonstrations or resulting from an optimization process can be used to help humans learn motor skills or to assist human movements. Potential applications of our results are human-robot collaboration, robot-assisted rehabilitation, robot-assisted movement training, teleoperation and other human-robot interaction tasks.

8.1 Summary of Contributions

This thesis gradually builds up a framework that can be used both to teach new motor skills to robots through human demonstrations ($H \rightarrow R$) as well as to enable robots to help humans learn motor skills or perform various kinds of motion ($R \rightarrow H$). Often there is the need of optimizing upon human demonstrations or adapting them to changes in the environment in order for the robot to solve a certain task. The robot can perform this optimization through trial and error, i.e. reinforcement learning ($R \circlearrowleft$). This thesis presents contributions to reinforcement learning and demonstrates how the human can profit from the optimized movements learned by the robot.
We present contributions to these three learning directions ($H \rightarrow R$, $R \circlearrowleft$ and $R \rightarrow H$) and show how they are tightly interconnected, building upon stochastic movement representations.

Humans Teaching Robots ($H \rightarrow R$)

In Chapter 2, a method to learn human-robot collaboration tasks from demonstrations is presented which can deal with missing data and that enables the robot to start reacting to human actions before the human finishes his/her movements. This method can substantially speed up human-robot collaboration in comparison to other state-of-the-art techniques.
Often a human demonstration is not enough to successfully teach a new motor skill to a robot because the human may be unable to give a suitable demonstration, the environment in the test phase can be different from the one experienced during the demonstration or the robot may be unable to generate the necessary forces and torques to accurately reproduce the demonstration. Chapter 3 presents an approach to address this problem. In this approach, the human provides demonstrations and incremental feedback to the robot, gradually leading it to perform the desired movements. Moreover, the robot can generalize the learned movements to different situations through probabilistic conditioning.

Robots Learning from Human Demonstrations and Improving through Trial and Error ($H \rightarrow R$ and $R \circlearrowleft$)

In some cases, there is the necessity of locally adapting the speed of execution of a movement without necessarily changing its shape in space. For example, when performing a golf putt, a fine adjustment in speed may be crucial. Chapter 4 shows that this fine adjustment in speed can be achieved by introducing parameters to a stochastic movement representation that explicitly model the progress of the movement in time and by using reinforcement learning to optimize these parameters.

Bidirectional Learning ($H \rightarrow R$, $R \circlearrowleft$ and $R \rightarrow H$)

Probability distributions of trajectories are basic components of our methods for imitation learning and reinforcement learning. Chapter 5 shows how these distributions can also be used to teach or assist humans. An algorithm proposed in that chapter aligns expert demonstrations in space and time to create a probability distribution of trajectories representing the demonstrated motor skill. Assuming that correct movements have high probability and incorrect movements have

low probability given this distribution, our algorithm can evaluate the movements of an apprentice and highlight any possible mistakes. Still in chapter 5, the same basic principle is used to give haptic feedback to the human, helping him/her accomplish a teleoperation task. We have performed user studies that demonstrated that the performance of users in the teleoperation task when assisted by a haptic device using our method was significantly higher than without assistance.
When expert demonstrations are not available or there is the need to adapt the available demonstrations to new situations, our method presented in Chapter 5 can improve upon the available demonstrations through reinforcement learning. Improving trajectories for assisted teleoperation in a setting like ours can be treated as a motion planning problem with multiple objectives depending, for example, on distances to objects of interest. For the purpose of solving this kind of problem while maintaining as much as possible of the variability of the original demonstrations, we have developed a new algorithm that explicitly takes into consideration the relevance of each trajectory parameter to each objective, which also achieves better convergence properties than other reinforcement learning methods which do not use the concept of relevance.
Chapter 6 presents a new reinforcement learning algorithm, which uses the connection between the concept of relevance and Pearson correlation. As opposed to computing the relevance in an iterative fashion such as in Chapter 5, this new algorithm can compute the relevance in one shot, which leads to a great speedup. Chapter 6 also shows how this algorithm can be used in conjunction with Gaussian Process regression to learn a mapping between environment configurations and suitable trajectory distributions to solve a given task. The ensuing framework can generalize movements to situations that are quite different from the situations experienced during demonstrations and can solve tasks in dynamic environments. This work represents a substantial improvement upon state-of-the-art research work in learning robot-assisted teleoperation from human demonstrations, since other proposed methods to not optimize upon the available human demonstrations. Furthermore, this work can greatly contribute to human-robot collaboration since the robot can constantly adapt to the movements of the human partner and to other changes in the environment.
Besides adapting to changes in the environment, an assistive system for teleoperation may need to recognize several possible solutions to a given task and adapt to changes in the intention of the human operator. Chapter 7 addresses these challenges by leveraging recent advances in variational inference to learn Gaussian mixture models.

8.2 Future Work

Ideas for future research based on the contributions of this thesis are presented in the following.

Extensions to Incremental Imitation Learning

Our method for incremental imitation learning of context-dependent motor skills 3 can at the moment be applied when the robot is not too heavy, does not move too fast and does not carry any dangerous objects. However, under these circumstances, it would be worth considering other ways for the human to give incremental feedback, such as teleoperation or through a graphical user interface.

Assisted Teleoperation in Unstructured Environments

We have applied our algorithms and frameworks to assist users in teleoperation tasks in relatively simple environments where the geometry of every object is known. In real teleoperation applications in unstructured environments such as exploring the interior of a building in a rescuing operation or in robot-assisted surgery, however, there may be many complicating factors. There may be unknown objects in the scene, some objects may not be visible, the dynamic properties of some objects may be unknown, etc. Endowing autonomous systems with the capability of dealing with unstructured environments would greatly extend the applicability of the algorithms and frameworks proposed in this thesis.
In chapter 6, an approach based on stochastic movement representations, reinforcement learning and Gaussian Process regression has been presented to optimize upon demonstrations and prior knowledge to deal with dynamic environments. The learning system has been initialized with demonstrations and prior knowledge about the task at hand. Subsequently, it has sampled many possible environment configurations and optimized its movements through trial and error to each of them. After each optimization, the mapping from environment state variables to movement parameters has been updated.
Dealing with the challenges of assisting teleoperation in unstructured environments could thus be in part addressed by scaling up the approach presented in chapter 6 and learning environment models based on interactions with real environments. The learning system would be required to learn a model of the environments it needs to interact with and update this model as more data becomes available. This model would then be used by the system to optimize its own

movements offline. New interactions with real environments would follow, which could again be used to optimize the system's world model and, subsequently, its own movements.
Depending on the requirements of the task, the environment model to be learned could be more or less simple. For example, if the objective of the task is just to avoid collisions with any objects in the environment, it may be unnecessary to model the dynamics properties of every object in the scene. Also in this case, if necessary to estimate the geometry of objects given images, this geometry could be in some cases simplified as long as it is suitable to avoid the objects.
Recent works have made important contributions towards learning environment models through interaction with the environment. In [106], deep learning is used to infer object properties such as mass, position, 3D shape and friction from videos or images. The proposed architecture can also predict the outcome of physical events with an accuracy comparable to the one achieved by human subjects. Agrawal et al. [107] propose a method to learn forward and inverse dynamics models from images of the environment and poking movements by the robot. Kansky et al. [108] propose a factor graph architecture to learn the dynamics of an environment from data. After being trained in a standard version of the Breakout Atari game, their architecture can achieve higher scores than state-of-the-art reinforcement learning algorithms also trained only in that same standard version.
Ideas such as the ones presented in these works could be used to learn models of the environment. The framework proposed in Chapter 6 could in turn be used to optimize movements in simulation using the learned environment models. Subsequently, the optimized movements could be used to assist a human in a teleoperation task, for example. The assisted teleoperation framework presented in Chapter 7 could also be used in conjunction with Gaussian Process regression or Neural Networks to provide an even faster adaptation to changes in the environment.

Extensions to Online Adaptation of Movements

In our work on online adaptation of Probabilistic Movement Primitives (ProMPs) presented in Chapter 6, the robot has to avoid colliding with a Pringles can. However, what if there are two or more Pringles cans in the scene?
Let us suppose the target of the robot is fixed and that there is only one Pringles can that needs to be avoided. Let us further suppose that training our learning system to avoid the Pringles can at N different positions in a grid results in successfully avoiding the Pringles can for any position on the table in the test phase.
By increasing the number of Pringles cans to two, the learning system could be trained with roughly N^2 different environment configurations (less than N^2 if we consider that two Pringles cans cannot be at the same position at the same time). Three Pringles would require roughly N^3 different environment configurations to train the learning system and so on. If a grid with 1000 points is used in the training to avoid one Pringles can, avoiding three Pringles cans would require training with approximately 10^9 different environment configurations, dramatically increasing the necessary amount of computation during training. Therefore, an important direction for future work is improving the scalability of this algorithm.
Perhaps, it is possible to blend Probabilistic Movement Primitives (ProMPs) which successfully avoid one Pringles can to achieve ProMPs that avoid multiple Pringles cans (See "Combination and blending of movement primitives" in [88]).
For many tasks, an agent does not need to take into consideration the whole environment at once. Therefore, it may be possible to train a learning agent using only a relatively small region around that agent, greatly reducing the amount of computation during training.
In Chapter 7, adaptation to changes in the environment has been achieved through an efficient implementation of a reinforcement learning algorithm based on variational inference. This adaptation could be sped up by the offline training of a mapping between environment and distribution of trajectories, as proposed in Chapter 6. This speedup is due to the fact that computing a distribution given the environment by using a regression model, e.g. Gaussian Process regression or Neural Network, is computationally less expensive than iteratively solving an optimization problem. Therefore, solving the scalability problem mentioned above would be beneficial despite efficient optimization.
Besides the scalability issue mentioned above, in order to make the frameworks presented in Chapters 6 and 7 applicable to more complex situations, it is important to prevent other types of collisions beyond collisions involving only the end effector of the robot. In our experiments where the robot has to avoid the Pringles can, only the collision between the robot hand and the Pringles can has been taken into consideration. In general, though, it would be necessary to take into consideration possible collisions between the robot arm and obstacles in the environment as well as self-collisions. Avoiding these types of collisions could be achieved by taking the geometry of the robot into consideration when designing the reward functions for our reinforcement learning algorithms.

Adapting Movements to the Dynamics of Objects

In the work presented in this thesis, kinematics plays an important role. The robot coordinates its movements in space and time with the movements of a human, it performs reaching tasks, it optimizes the speed profile of movements, it gives

visual and haptic feedback to the user concerning the position and/or orientation along a trajectory, it avoids obstacles, etc.

Nevertheless, our approach could also be applied in situations where dynamics plays an important role. For example, the robot may have to move a bottle from a start position to an end position along a certain trajectory. The joint torques necessary to perform this movement may vary considerably depending on the amount of liquid inside the bottle.

In our work, a probability distribution of trajectories is computed given the action of the human (Chapter 2) or given environment configurations (Chapters 3 and 6). The same principle could be used to manipulate objects with different dynamics. Provided a dataset of reaction forces generated by lifting a bottle and the sequence of joint torques at each time step to track certain trajectories, it is possible to compute a probability distribution of joint torques given reaction force and desired trajectory. This could be achieved for example through multivariable Gaussian conditioning as in Chapters 2 and 3, potentially using a mixture of Gaussians as in [37] to deal with nonlinear correlations, or through Gaussian Process regression as in Chapter 6.

8.3 Research Papers Included in this Thesis

The research presented in this thesis has led to the following research papers:

Ewerton, M.; Maeda, G.J.; Peters, J.; Neumann, G. (2015). Learning Motor Skills from Partially Observed Movements Executed at Different Speeds, *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pp.456–463.

Ewerton, M.; Maeda, G.J.; Kollegger, G.; Wiemeyer, J.; Peters, J. (2016). Incremental Imitation Learning of Context-Dependent Motor Skills, *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, pp.351–358.

Ewerton, M.; Maeda, G.; Neumann, G.; Kisner, V.; Kollegger, G.; Wiemeyer, J.; Peters, J. (2016). Movement Primitives with Multiple Phase Parameters, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp.201–206.

Ewerton, M.; Rother, D.; Weimar, J.; Kollegger, G.; Wiemeyer, J.; Peters, J.; Maeda, G. (2018). Assisting Movement Training and Execution with Visual and Haptic Feedback, *Frontiers in Neurorobotics*.

Ewerton, M.; Maeda, G.; Koert, D.; Kolev, Z.; Takahashi, M.; Peters, J. (2019). Learning Trajectory Distributions for Assisted Teleoperation and Path Planning, *Frontiers in Robotics and AI, section Robotic Control Systems (submitted)*.

Ewerton, M.; Arenz, O.; Peters, J. (2019). Assisted Teleoperation in Changing Environments with a Mixture of Virtual Guides, *Advanced Robotics (submitted)*.

Bibliography

[1] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[3] J. Solis, C. A. Avizzano, and M. Bergamasco, "Teaching to write japanese characters using a haptic interface," in *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on*, pp. 255–262, IEEE, 2002.

[4] T. Tsumugiwa, R. Yokogawa, and K. Hara, "Variable impedance control based on estimation of human arm stiffness for human-robot cooperative calligraphic task," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1, pp. 644–650, IEEE, 2002.

[5] H. Yin, A. Billard, and A. Paiva, "Bidirectional learning of handwriting skill in human-robot interaction," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*, pp. 243–244, ACM, 2015.

[6] G. Rauter, N. Gerig, R. Sigrist, R. Riener, and P. Wolf, "When a robot teaches humans: Automated feedback selection accelerates motor learning," *Science Robotics*, vol. 4, no. 27, p. eaav1560, 2019.

[7] L. B. Rosenberg, "The use of virtual fixtures as perceptual overlays to enhance operator performance in remote environments.," tech. rep., DTIC Document, 1992.

[8] A. A. Timmermans, H. A. Seelen, R. D. Willmann, and H. Kingma, "Technology-assisted training of arm-hand skills in stroke: concepts on reacquisition of motor control and therapist guidelines for rehabilitation technology design," *Journal of neuroengineering and rehabilitation*, vol. 6, no. 1, 2009.

[9] M. O. Ernst and M. S. Banks, "Humans integrate visual and haptic information in a statistically optimal fashion," *Nature*, vol. 415, no. 6870, pp. 429–433, 2002.

[10] J. Kümmel, A. Kramer, and M. Gruber, "Robotic guidance induces long-lasting changes in the movement pattern of a novel sport-specific motor task," *Human movement science*, vol. 38, pp. 23–33, 2014.

[11] J. N. Chopp, S. L. Fischer, and C. R. Dickerson, "The impact of work configuration, target angle and hand force direction on upper extremity muscle activity during sub-maximal overhead work," *Ergonomics*, vol. 53, no. 1, pp. 83–91, 2010.

[12] B. P. Bernard and V. Putz-Anderson, "Musculoskeletal disorders and workplace factors; a critical review of epidemiologic evidence for work-related musculoskeletal disorders of the neck, upper extremity, and low back," *U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES*, 1997.

[13] P. P. F. Kuijer, J. H. Verbeek, A. Seidler, R. Ellegast, C. T. Hulshof, M. H. Frings-Dresen, and H. F. Van der Molen, "Work-relatedness of lumbosacral radiculopathy syndrome: Review and dose-response meta-analysis," *Neurology*, vol. 91, no. 12, pp. 558–564, 2018.

[14] J. Verbeek, C. Mischke, R. Robinson, S. Ijaz, P. Kuijer, A. Kievit, A. Ojajärvi, and K. Neuvonen, "Occupational exposure to knee loading and the risk of osteoarthritis of the knee: a systematic review and a dose-response meta-analysis," *Safety and health at work*, vol. 8, no. 2, pp. 130–142, 2017.

[15] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

[16] "Human-robot collaboration with KUKA robots." `https://www.kuka.com/en-de/technologies/human-robot-collaboration`, Feb. 2019.

[17] L. Probst, L. Frideres, B. Pedersen, and C. Caputi, "Service innovation for smart industry: human–robot collaboration," *European Commission, Luxembourg*, 2015.

[18] "5 Medical Robots Making a Difference in Healthcare." https://online-engineering.case.edu/blog/medical-robots-making-a-difference, Dec. 2017.

[19] "The daVinci® Surgical Robot." https://youtu.be/C17-bGquIjI, Mar. 2009.

[20] R. H. Taylor, A. Menciassi, G. Fichtinger, P. Fiorini, and P. Dario, "Medical robotics and computer-integrated surgery," in *Springer handbook of robotics*, pp. 1657–1684, Springer, 2016.

[21] M. Iosa, G. Morone, A. Cherubini, and S. Paolucci, "The three laws of neurorobotics: a review on what neurorehabilitation robots should do for patients and clinicians," *Journal of Medical and Biological Engineering*, vol. 36, no. 1, pp. 1–11, 2016.

[22] P. Morasso, M. Casadio, P. Giannoni, L. Masia, V. Sanguineti, V. Squeri, and E. Vergaro, "Desirable features of a "humanoid" robot-therapist," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 2418–2421, IEEE, 2009.

[23] K. Muelling, A. Venkatraman, J.-S. Valois, J. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell, "Autonomy infused teleoperation with application to bci manipulation," *arXiv preprint arXiv:1503.05451*, 2015.

[24] H. M. Van der Loos, D. J. Reinkensmeyer, and E. Guglielmelli, "Rehabilitation and health care robotics," in *Springer handbook of robotics*, pp. 1685–1728, Springer, 2016.

[25] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006.

[26] I. Havoutis and S. Calinon, "Learning assistive teleoperation behaviors from demonstration," in *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*, pp. 258–263, IEEE, 2016.

[27] F. Abi-Farraj, T. Osa, N. P. J. Peters, G. Neumann, and P. R. Giordano, "A learning-based shared control architecture for interactive task execution," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 329–335, IEEE, 2017.

[28] I. Havoutis and S. Calinon, "Supervisory teleoperation with online learning and optimal control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, IEEE, 2017.

[29] "This is a robot. It can also teach you how to play golf." https://www.golfdigest.com/story/this-is-a-robot-it-can-also-te, Jan. 2014.

[30] "RoboGolfPro Offers An Instant Golf Swing Fix." https://www.forbes.com/sites/scottkramer/2017/08/22/robogolfpro-offers-an-instant-golf-swing-fix/#542f7b941a3b, Aug. 2017.

[31] Y. Kowsar, M. Moshtaghi, E. Velloso, L. Kulik, and C. Leckie, "Detecting unseen anomalies in weight training exercises," in *Proceedings of the 28th Australian Conference on Computer-Human Interaction*, pp. 517–526, ACM, 2016.

[32] G. I. Parisi, S. Magg, and S. Wermter, "Human motion assessment in real time using recurrent self-organization," in *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*, pp. 71–76, IEEE, 2016.

[33] R. Sigrist, G. Rauter, R. Riener, and P. Wolf, "Augmented visual, auditory, haptic, and multimodal feedback in motor learning: a review," *Psychonomic bulletin & review*, vol. 20, no. 1, pp. 21–53, 2013.

[34] M. Hamaya, T. Matsubara, T. Noda, T. Teramae, and J. Morimoto, "Learning assistive strategies for exoskeleton robots from user-robot physical interaction," *Pattern Recognition Letters*, vol. 99, pp. 67–76, 2017.

[35] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 2616–2624, 2013.

[36] G. Maeda, M. Ewerton, R. Lioutikov, H. Ben Amor, J. Peters, and G. Neumann, "Learning interaction for collaborative tasks with probabilistic movement primitives," in *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2014.

[37] M. Ewerton, G. Neumann, R. Lioutikov, H. Ben Amor, J. Peters, and G. Maeda, "Learning multiple collaborative tasks with a mixture of interaction primitives," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2015.

[38] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[39] H. Ben Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human-robot cooperation tasks," in *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[40] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, 1978.

[41] S. Calinon, E. L. Sauser, A. G. Billard, and D. G. Caldwell, "Evaluation of a probabilistic approach to learn and reproduce gestures by imitation," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2671–2676, IEEE, 2010.

[42] P. Englert and M. Toussaint, "Reactive phase and task space adaptation for robust motion execution," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 109–116, IEEE, 2014.

[43] R. Vuga, B. Nemec, and A. Ude, "Speed profile optimization through directed explorative learning," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pp. 547–553, IEEE, 2014.

[44] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *Proceedings of the 25th international conference on Machine learning*, pp. 144–151, ACM, 2008.

[45] J. Van Den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2074–2081, IEEE, 2010.

[46] F. Meier, E. Theodorou, F. Stulp, and S. Schaal, "Movement segmentation using a primitive library," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3407–3412, IEEE, 2011.

[47] C. M. Bishop *et al.*, *Pattern recognition and machine learning*, vol. 1. springer New York, 2006.

[48] T. Lens and O. von Stryk, "Design and dynamics model of a lightweight series elastic tendon-driven robot arm," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 4512–4518, IEEE, 2013.

[49] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *International Journal of Robotics Research*, vol. 31, no. 3, pp. 330–345, 2012.

[50] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pp. 255–262, ACM, 2007.

[51] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, no. 2-3, pp. 115–131, 2011.

[52] A. Jain, B. Wojcik, T. Joachims, and A. Saxena, "Learning trajectory preferences for manipulators via iterative improvement," in *Advances in Neural Information Processing Systems*, pp. 575–583, 2013.

[53] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, pp. 391–398, ACM, 2012.

[54] B. D. Argall, E. L. Sauser, and A. G. Billard, "Tactile guidance for policy refinement and reuse," in *Development and Learning (ICDL), 2010 IEEE 9th International Conference on*, pp. 7–12, IEEE, 2010.

[55] E. L. Sauser, B. D. Argall, G. Metta, and A. G. Billard, "Iterative learning of grasp adaptation through human corrections," *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 55–71, 2012.

[56] S. Calinon, T. Alizadeh, and D. G. Caldwell, "On improving the extrapolation capability of task-parameterized movement models," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 610–616, IEEE, 2013.

[57] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[58] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[59] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.

[60] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[61] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[62] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 2, pp. 286–298, 2007.

[63] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[64] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Proceedings of the 16th International Symposium on Robotics Research (ISRR)*, 2013.

[65] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2, pp. 1398–1403, IEEE, 2002.

[66] J. Kober, K. Mulling, O. Kromer, C. Lampert, B. Scholkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pp. 853–858, IEEE, 2010.

[67] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, pp. 423–428, IEEE, 2013.

[68] S. Kim, E. Gribovskaya, and A. Billard, "Learning motion dynamics to catch a moving object," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pp. 106–111, IEEE, 2010.

[69] K. Kronander, M. S. Khansari-Zadeh, and A. Billard, "Learning to control planar hitting motions in a minigolf-like task," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 710–717, IEEE, 2011.

[70] M. Ewerton, G. Maeda, J. Peters, and G. Neumann, "Learning motor skills from partially observed movements executed at different speeds," in *Accepted: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[71] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, ACM, 2007.

[72] C. J. Wu, "On the convergence properties of the em algorithm," *The Annals of statistics*, pp. 95–103, 1983.

[73] G. Raiola, X. Lamy, and F. Stulp, "Co-manipulation with multiple probabilistic virtual guides," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 7–13, IEEE, 2015.

[74] H. Soh and Y. Demiris, "Learning assistance by demonstration: Smart mobility with shared control and paired haptic controllers," *Journal of Human-Robot Interaction*, vol. 4, no. 3, pp. 76–100, 2015.

[75] J. Listgarten, R. M. Neal, S. T. Roweis, and A. Emili, "Multiple alignment of continuous time series," in *Advances in neural information processing systems*, pp. 817–824, 2004.

[76] P. Sanguansat, "Multiple multidimensional sequence alignment using generalized dynamic time warping," *WSEAS Transactions on Mathematics*, vol. 11, no. 8, pp. 668–678, 2012.

[77] C. Goodall, "Procrustes methods in the statistical analysis of shape," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 285–339, 1991.

[78] B. Bocsi, L. Csató, and J. Peters, "Alignment-based transfer learning for robot models," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–7, IEEE, 2013.

[79] R. M. Holladay and S. S. Srinivasa, "Distance metrics and algorithms for task space path optimization," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 5533–5540, IEEE, 2016.

[80] N. Makondo, B. Rosman, and O. Hasegawa, "Knowledge transfer for learning robot models via local procrustes analysis," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pp. 1075–1082, IEEE, 2015.

[81] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 48–55, IEEE, 2009.

[82] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, "Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks," *Autonomous Robots*, pp. 1–20, 2016.

[83] M. Ewerton, G. Maeda, G. Neumann, V. Kisner, G. Kollegger, J. Wiemeyer, and J. Peters, "Movement primitives with multiple phase parameters," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 201–206, IEEE, 2016.

[84] M. Ewerton, D. Rother, J. Weimar, G. Kollegger, J. Wiemeyer, J. Peters, and G. Maeda, "Assisting movement training and execution with visual and haptic feedback," *Frontiers in neurorobotics*, vol. 12, p. 24, 2018.

[85] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 489–494, IEEE, 2009.

[86] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4569–4574, IEEE, 2011.

[87] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2397–2403, IEEE, 2010.

[88] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[89] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, "Guiding trajectory optimization by demonstrated distributions," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 819–826, 2017.

[90] D. Koert, G. Maeda, R. Lioutikov, G. Neumann, and J. Peters, "Demonstration based trajectory optimization for generalizable robot motions," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 515–522, IEEE, 2016.

[91] M. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Intelligent robots and systems*, 2018.

[92] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[93] R. Bloss, "How do you decommission a nuclear installation? call in the robots," *Industrial Robot: An International Journal*, vol. 37, no. 2, pp. 133–136, 2010.

[94] I. Havoutis and S. Calinon, "Learning from demonstration for semi-autonomous teleoperation," *Autonomous Robots*, pp. 1–14, 2018.

[95] F. Abi-Farraj, C. Pacchierotti, O. Arenz, G. Neumann, and P. R. Giordano, "A haptic shared-control architecture for guided multi-target robotic grasping," *IEEE transactions on haptics*, 2019.

[96] O. Arenz, M. Zhong, G. Neumann, *et al.*, "Efficient gradient-free variational inference using policy search," in *Proceedings of the thirty-fifth international conference on Machine learning*, 2018.

[97] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004.

[98] P. Aigner and B. McCarragher, "Human integration into robot control utilising potential fields," in *Proceedings of International Conference on Robotics and Automation*, vol. 1, pp. 291–296, IEEE, 1997.

[99] A. M. Howard and C. H. Park, "Haptically guided teleoperation for learning manipulation tasks," in *RSS 2007 Workshop on Manipulation for Human Environments*, Georgia Institute of Technology, 2007.

[100] G. Gioioso, M. Mohammadi, A. Franchi, and D. Prattichizzo, "A force-based bilateral teleoperation framework for aerial robots in contact with the environment," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 318–324, IEEE, 2015.

[101] L. Huber, A. Billard, and J.-J. Slotine, "Avoidance of convex and concave obstacles with convergence ensured through contraction," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1462–1469, 2019.

[102] S. M. Khansari-Zadeh and O. Khatib, "Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors," *Autonomous Robots*, vol. 41, no. 1, pp. 45–69, 2017.

[103] D. Koert, S. Trick, M. Ewerton, M. Lutter, and J. Peters, "Online learning of an open-ended skill library for collaborative tasks," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pp. 1–9, IEEE, 2018.

[104] P. M. Engel and M. R. Heinen, "Incremental learning of multivariate gaussian mixture models," in *Brazilian Symposium on Artificial Intelligence*, pp. 82–91, Springer, 2010.

[105] E. Pignat, T. Lembono, and S. Calinon, "Variational inference with mixture model approximation: Robotic applications," *arXiv preprint arXiv:1905.09597*, 2019.

[106] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," in *Advances in neural information processing systems*, pp. 127–135, 2015.

[107] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems*, pp. 5074–5082, 2016.

[108] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1809–1818, JMLR. org, 2017.

A Publication List

Journal Papers

Ewerton, M.; Rother, D.; Weimar, J.; Kollegger, G.; Wiemeyer, J.; Peters, J.; Maeda, G. (2018). Assisting Movement Training and Execution with Visual and Haptic Feedback, *Frontiers in Neurorobotics*.

Maeda, G.; Neumann, G.; **Ewerton, M.**; Lioutikov, R.; Kroemer, O.; Peters, J. (2017). Probabilistic Movement Primitives for Coordination of Multiple Human-Robot Collaborative Tasks, *Autonomous Robots (AURO)*, 41, 3, pp.593-612.

Maeda, G.; **Ewerton, M.**; Neumann, G.; Lioutikov, R.; Peters, J. (2017). Phase Estimation for Fast Action Recognition and Trajectory Generation in Human-Robot Collaboration, *International Journal of Robotics Research (IJRR)*, 36, 13-14, pp.1579-1594.

Dermy, O.; Paraschos, A.; **Ewerton, M.**; Charpillet, F.; Peters, J.; Ivaldi, S (2017). Prediction of intention during interaction with iCub with Probabilistic Movement Primitives, *Frontiers in Robotics and AI*, 4, pp.45.

Kollegger, G.; **Ewerton, M.**; Wiemeyer, J.; Peters, J. (2017). BIMROB – Bidirectional Interaction Between Human and Robot for the Learning of Movements, in: Lames, M.; Saupe, D.; Wiemeyer, J. (eds.), *Proceedings of the 11th International Symposium on Computer Science in Sport (IACSS 2017)*, pp.151–163, Springer International Publishing.

Maeda, G.; **Ewerton, M.**; Koert, D.; Peters, J. (2016). Acquiring and Generalizing the Embodiment Mapping from Human Observations to Robot Skills, *IEEE Robotics and Automation Letters (RA-L)*, 1, 2, pp.784–791.

Conference Papers

Koert, D.; Trick, S.; **Ewerton, M.**; Lutter, M.; Peters, J. (2018). Online Learning of an Open-Ended Skill Library for Collaborative Tasks, *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*.

Kollegger, G.; Götz, F.; **Ewerton, M.**; Wiemeyer, J.; Peters, J. (2018). Einfluss der Beobachtungsperspektive beim Bewegungslernen von Mensch-Roboter-Dyaden, in: M. Lames, D. Link & V. Senner (eds.), *12. Symposium der dvs-Sektion Sportinformatik und Sporttechnologie*, pp.51-52.

Ewerton, M.; Kollegger, G.; Maeda, G.; Wiemeyer, J.; Peters, J. (2017). Iterative Feedback-basierte Korrekturstrategien beim Bewegungslernen von Mensch-Roboter-Dyaden, *DVS Sportmotorik 2017*.

Kollegger, G.; Reinhardt, N.; **Ewerton, M.**; Peters, J.; Wiemeyer, J. (2017). Die Bedeutung der Beobachtungsperspektive beim Bewegungslernen von Mensch-Roboter-Dyaden, *DVS Sportmotorik 2017*.

Wiemeyer, J.; Peters, J.; Kollegger, G.; **Ewerton, M.** (2017). BIMROB – Bidirektionale Interaktion von Mensch und Roboter beim Bewegungslernen, *DVS Sportmotorik 2017*.

Kollegger, G.; **Ewerton, M.**; Wiemeyer, J.; Peters, J. (2017). BIMROB – Bidirectional Interaction between human and robot for the learning of movements – Robot trains human – Human trains robot, *23. Sportwissenschaftlicher Hochschultag der dvs*.

Maeda, G.; **Ewerton, M.**; Osa, T.; Busch, B.; Peters, J. (2017). Active Incremental Learning of Robot Movement Primitives, *Proceedings of the Conference on Robot Learning (CoRL)*.

Kollegger, G.; Wiemeyer, J.; **Ewerton, M.**; Peters, J. (2017). BIMROB - Bidirectional Interaction between human and robot for the learning of movements - Robot trains human - Human trains robot, in: A. Schwirtz, F. Mess, Y. Demetriou & V. Senner (eds.), *Inovation & Technologie im Sport - 23. Sportwissenschaftlicher Hochschultag der deutschen Vereinigung für Sportwissenschaft*, pp.179, Czwalina-Feldhaus.

Ewerton, M.; Maeda, G.; Neumann, G.; Kisner, V.; Kollegger, G.; Wiemeyer, J.; Peters, J. (2016). Movement Primitives with Multiple Phase Parameters, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp.201–206.

Maeda, G.; Maloo, A.; **Ewerton, M.**; Lioutikov, R.; Peters, J. (2016). Anticipative Interaction Primitives for Human-Robot Collaboration, *AAAI Fall Symposium Series. Shared Autonomy in Research and Practice*, Arlington, VA, USA.

Ewerton, M.; Maeda, G.J.; Kollegger, G.; Wiemeyer, J.; Peters, J. (2016). Incremental Imitation Learning of Context-Dependent Motor Skills, *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, pp.351–358.

Kollegger, G.; **Ewerton, M.**; Peters, J.; Wiemeyer, J. (2016). Bidirektionale Interaktion zwischen Mensch und Roboter beim Bewegungslernen (BIMROB), *11. Symposium der DVS Sportinformatik*.

Ewerton, M.; Neumann, G.; Lioutikov, R.; Ben Amor, H.; Peters, J.; Maeda, G. (2015). Learning Multiple Collaborative Tasks with a Mixture of Interaction Primitives, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp.1535–1542.

Ewerton, M.; Maeda, G.J.; Peters, J.; Neumann, G. (2015). Learning Motor Skills from Partially Observed Movements Executed at Different Speeds, *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pp.456–463.

Maeda, G.; Neumann, G.; **Ewerton, M.**, Lioutikov, R.; Peters, J. (2015). A Probabilistic Framework for Semi-Autonomous Robots Based on Interaction Primitives with Phase Estimation, *Proceedings of the International Symposium of Robotics Research (ISRR)*.

Maeda, G.J.; **Ewerton, M.**; Lioutikov, R.; Amor, H.B.; Peters, J.; Neumann, G. (2014). Learning Interaction for Collaborative Tasks with Probabilistic Movement Primitives, *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, pp.527–534.

Ben Amor, H.; Vogt, D.; **Ewerton, M.**; Berger, E.; Jung, B.; Peters, J. (2013). Learning Responsive Robot Behavior by Imitation, *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Kroemer, O.; Ben Amor, H.; **Ewerton, M.**; Peters, J. (2012). Point Cloud Completion Using Extrusions, *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*.

Workshop Papers

Ewerton, M.; Maeda, G.; Rother, D.; Weimar, J.; Lotter, L.; Kollegger, G.; Wiemeyer, J.; Peters, J. (2017). Assisting the practice of motor skills by humans with a probability distribution over trajectories, *Workshop Human-in-the-loop robotic manipulation: on the influence of the human role at IROS 2017*, Vancouver, Canada.

Maeda, G.; Maloo, A.; **Ewerton, M.**; Lioutikov, R.; Peters, J. (2016). Proactive Human-Robot Collaboration with Interaction Primitives, *International Workshop on Human-Friendly Robotics (HFR)*, Genoa, Italy.

Ewerton, M.; Neumann, G.; Lioutikov, R.; Ben Amor, H.; Peters, J.; Maeda, G. (2015). Modeling Spatio-Temporal Variability in Human-Robot Interaction with Probabilistic Movement Primitives, *Workshop on Machine Learning for Social Robotics, ICRA*.

B Curriculum Vitae

Current Position

Since Jan. 2015	Ph.D. Student at IAS TU Darmstadt. I am a Ph.D. Student at the Intelligent Autonomous Systems group of the TU Darmstadt. The topic of my research project is "Bidirectional Human-Robot Learning: Imitation and Skill Improvement". I investigate how humans and robots can improve their movements by interacting with each other. Supervisors: Prof. Dr. Jan Peters and Dr. Guilherme Maeda.

Research Interests

Artificial Intelligence, Machine Learning, Robotics, Imitation Learning, Human-Robot Interaction, Motor Skill Learning

Education Background

Oct. 2012–Dec. 2014	M.Sc. in Electrical Engineering and Information Technology (Elektrotechnik und Informationstechnik), option Technical Computer Science (Datentechnik). Technische Universität Darmstadt, Germany. Thesis: "Modeling Human-Robot Interaction with Probabilistic Movement Representations". Supervisors: Prof. Dr. Jan Peters, Prof. Dr.-Ing. Klaus Hofmann, Prof. Dr. techn. Gerhard Neumann and Dr. Guilherme Maeda.
Oct. 2009–Sep. 2012	B.Sc. in Electrical Engineering and Information Technology (Elektrotechnik und Informationstechnik), option Technical Computer Science (Datentechnik). Technische Universität Darmstadt, Germany. Thesis: "Job Characteristics on Crowdsourcing Platforms". Supervisors: Prof. Dr.-Ing. Ralf Steinmetz, Dr.-Ing. Philipp Scholl and M.Sc. Sebastian Schmidt.
Apr. 2009–Sep. 2009	Exchange Student of Electrical Engineering and Information Technology (Elektrotechnik und Informationstechnik), option Technical Computer Science (Datentechnik) at the TU Darmstadt.
Apr. 2008–Mar. 2009	Exchange Student of Computer Science at the TU Darmstadt.
Mar. 2005–Dec. 2007	Bachelor Student of Computer Engineering at the University of São Paulo (USP), Brazil.

Previous Work Experience

Apr. 2012–Dec. 2013	Work at IAS TU Darmstadt. I worked as a research assistant at the Intelligent Autonomous Systems group at the TU Darmstadt. There my main topics were "3D Reconstruction from multiple Kinect cameras" and "Human-Robot Interaction".
June 2009–Mar. 2011	Work at Fraunhofer IGD. I worked as a research assistant at the department "Information Visualization and Visual Analytics" at the Fraunhofer IGD in Darmstadt. As part of my activities, I wrote ontologies, programmed in ActionScript and JAVA and executed a number of tasks involving Semantics and Semantics Visualization.
Oct. 2006–Aug. 2007	Work at interlab USP. I worked as a volunteer at the laboratory for interactive technologies (interlab) of the University of São Paulo (USP), Brazil. There I worked on the development of a soccer game for two robots using JAVA (https://www.youtube.com/watch?v=MrXsK54NjIc&feature=related).

Teaching Assistance

Winter Semester 2017/2018	Robot Learning. TU Darmstadt.
Summer Semester 2017	Statistical Machine Learning. TU Darmstadt.
Winter Semester 2016/2017	Robot Learning Integrated Project. TU Darmstadt.

Supervision of Students

2018	Zlatko Kolev. *Joint Learning of Humans and Robots*. Bachelor Thesis.
2017–2018	Michael Burkhardt, Moritz Knaust and Susanne Trick. *From Robots to Cobots*. Robot Learning Integrated Project.
2017	Claudia Lölkes. *Incremental Imitation Learning with Estimation of Uncertainty*. Bachelor Thesis.
2016–2017	David Rother, Jakob Weimar and Lars Lotter. *Teaching People how to Write Japanese Characters*. Robot Learning Integrated Project.
2015–2016	Florian Brandherm. *Learning Minigolf with the BioRob*. Robot Learning Integrated Project.

Honors and Awards

2015	Best Conference Paper Award - Finalist for the paper “Learning multiple collaborative tasks with a mixture of interaction primitives” jointly with Gerhard Neumann, Rudolf Lioutikov, Heni Ben Amor, Jan Peters, Guilherme Jorge Maeda
2015	Best Student Conference Paper Award - Finalist for the paper “Learning multiple collaborative tasks with a mixture of interaction primitives” jointly with Gerhard Neumann, Rudolf Lioutikov, Heni Ben Amor, Jan Peters, Guilherme Jorge Maeda
2015	Service Robotics Best Paper Award - Finalist for the paper “Learning multiple collaborative tasks with a mixture of interaction primitives” jointly with Gerhard Neumann, Rudolf Lioutikov, Heni Ben Amor, Jan Peters, Guilherme Jorge Maeda

Reviewing

2019	Intelligent Service Robotics (JIST)
2019	Autonomous Robots (AURO)
2019	IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
2019	IEEE Transactions on Robotics (T-RO)
2019	IEEE Transactions on Cybernetics
2019	IEEE International Conference on Robotics and Automation (ICRA)
2018	Workshop on Reinforcement Learning under Partial Observability, NIPS
2018	IEEE Robotics and Automation Letters (RA-L)
2018	Conference on Robot Learning (CoRL)
2018	Journal of Intelligent & Robotic Systems
2018	IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
2018	International Journal of Robotics Research (IJRR)
2018	Autonomous Robots (AURO)
2018	IEEE International Conference on Robotics and Automation (ICRA)
2017	IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)
2017	IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
2017	Autonomous Robots (AURO)
2016	IEEE/RAS International Conference on Humanoid Robots (HUMANOIDS)
2016	IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
2015	IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
2015	IEEE Robotics and Automation Magazine (RAM)
2015	IEEE Transactions on Robotics (T-RO)
2014	IEEE Transactions on Robotics (T-RO)

Invited Talks

Feb. 07, 2019	University of Tokyo, Tokyo, Host: Takayuki Osa
Feb. 04, 2019	Advanced Telecommunications Research Institute International (ATR), Kyoto, Host: Guilherme Maeda
Jan. 29, 2019	Honda Research Institute Japan Co-Research Lab, Tokyo, Host: Chris Garry
Jan. 22, 2019	Keio University, Yokohama, Host: Masaki Takahashi
Nov. 07, 2017	Symposium “Kollaborative Robotik”. Organizers: IHK Hessen innovativ, VDI Bezirksverein Mittelhessen e.V. and VDE Rhein-Main. Location: Geschäftsstelle Gießen Industrie- und Handelskammer Gießen-Friedberg

Research Stays

Jan. 2019–Mar. 2019	Keio University, School of Science and Engineering, Department of System Design Engineering, Takahashi Laboratory, Yokohama, Japan

Summer School Attendance

2015	Machine Learning Summer School at the Max Planck Institute for Intelligent Systems, Tübingen, Germany

Languages

Portuguese	Mother tongue
English	Fluent
German	Fluent (Goethe-Certificate-C1, TestDaf, DSH 2)